ADA010232

EPRF Technical Report
2-75 (CSC)

FEASIBILITY STUDY OF A QUADRILATERALIZED SPHERICAL CUBE EARTH DATA BASE

F.K. Chan

E.M. O'Neill

COMPUTER SCIENCES CORPORATION
SYSTEM SCIENCES DIVISION
8728 Colesville Road
Silver Spring, Maryland 20910

MARCH 1975

Final Report

Approved For Public Release:
Distribution Unlimited

Prepared for

COMMANDER, NAVAL AIR SYSTEMS COMMAND
Department of the Navy
Washington, D.C. 20361

ENVIRONMENTAL PREDICTION RESEARCH FACILITY
Naval Postgraduate School
Monterey, California 93940

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER EPRF Technical Report 2-75(CSC)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD A010232
4. TITLE (and Subtitle) Feasibility Study of a Quadrilateralized Spherical Cube Earth Data Base		5. TYPE OF REPORT & PERIOD COVERED Final Report May 1974 - March 1975
		6. PERFORMING ORG. REPORT NUMBER CSC/TR-75/6007
7. AUTHOR(s) F. K. Chan E. M. O'Neill		8. CONTRACT OR GRANT NUMBER(s) N66314-74-C-1340
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation System Sciences Division 8728 Colesville Rd., Silver Spring, Md. 20910		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS EPRF WU 114:6-1
11. CONTROLLING OFFICE NAME AND ADDRESS Environmental Prediction Research Facility Naval Postgraduate School Monterey, California 93940		12. REPORT DATE April 14, 1975
		13. NUMBER OF PAGES 136
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release:Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

PRICES SUBJECT TO CHANGE

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Base, Meteorology, Map Projection, Equal Area Projection, Array Mapping, Satellite Data Base

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the results of research into the feasibility of storing satellite meteorological data in a high-resolution, equal-area, computer-accessible data base. Equal area elements are stored in square arrays representing the faces of a cube inscribed within the Earth. The required area-preserving transformation is presented along with its inverse. A unique array-mapping scheme is presented, which preserves near-neighbor relations and allows rapid index computation. The implementation of this data base on rotational storage devices is discussed, and calculated execution times are presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

ACKNOWLEDGMENT

The research effort described in this report was made possible by the U.S. Navy Environmental Prediction Research Facility (EPRF), Monterey, California.

The authors wish to specifically acknowledge the thought-provoking sessions provided by Mr. R. Nagle, Head, Satellite Applications Department, EPRF. During these sessions with the authors, Mr. Nagle contributed inputs for various aspects of the development of the pilot model, and also provided guidelines toward the ultimate operational use to which this model would be applied.

EXECUTIVE SUMMARY

This report presents the results of a feasibility study of the Quadrilateralized Spherical Cube Earth Data Base. The structure of this data base is derived from a spherical cube, which is obtained by radially projecting a cube onto its circumscribing sphere. An appropriate set of curvilinear coordinates is chosen such that the resolution cells on the spherical cube are of equal area and are also of essentially the same shape. This geometrical construction eliminates both the necessity to account for nonequal-area cells, and the need to compute the location of every cell in the data base.

The main properties of the Earth data base are as follows:

1. The indexing scheme is binary in nature, and telescopic in the sense that each additional level of resolution is addressed by appending additional binary bits. Thus, minimal work is needed for indexing cells of higher resolution.

2. The resolution cells are strung together in a two-dimensional manner, so as to accomplish area coverage with a serial bit string. Consequently, a higher degree of proximity is achieved for near-neighbors in this stringing pattern than in the usual one-dimensional array of stringing by rows and columns.

3. The cell addresses are readily computed because of the indexing scheme which is the same regardless of the resolution level, and because of the stringing pattern which permits the decomposition of the cell address into two independent binary indices.

4. The conversion from geographic coordinates to data base coordinates is comparatively simple because of the simplicity of the data base structure.

5. Incoming data can be stored rapidly by interpolation, using benchmarks only occasionally. This method of fast-filling is made possible by the equal-area nature and translational shape invariance of the data base resolution cells.

6. Input/output operations with this data base are also simplified because of the rectangularized nature of the data base records and the rhombic nature of the interpolation blocks.

7. The user can rapidly and directly access data corresponding to specified geographic regions of arbitrary shape and size. This data-accessing is accomplished by retrieving the relatively few bit-strings which lie within the associated data base records. The rapidity and directness of data access are the result of equal-area resolution, translational invariance, indexing scheme, stringing pattern, and relatively simple coordinate transformation.

8. The primary contemplated uses of the retrieved data are mathematical computations and visual display. For the former, the equal-area resolution property eliminates the need to distinguish between density measurements and integrated measurements. For the latter, the quadrilateralized nature of the resolution cells on the spherical cube and the comparative simplicity of coordinate transformation both simplify and minimize the internal operations.

Based on numerical results obtained in this study, it is concluded that implementation of the proposed Quadrilateralized Spherical Cube Earth Data Base is feasible on current large-scale computer systems.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

SECTION 1 - INTRODUCTION

In the numerous satellites presently orbiting the Earth, enormous amounts of data are continuously taken of the Earth's surface and atmosphere. These data are of a varied nature: topography, crop distribution, sea surface temperature, cloud coverage, etc. The measurements are used by research and applications personnel of diverse scientific disciplines. These users usually employ and Earth-oriented coordinate system, such as the traditional geographic frame of reference. Thus, it is not surprising that almost all existing Earth data bases have been constructed with latitudes and longitudes as gridlines, either in a patched-up partial fashion or in the entire outlay.

However, what is convenient to the user is not necessarily also efficient from the standpoint of data management and data processing by the computer. Efficiency is especially important because of the large amounts of data rapidly acquired in global coverage, the necessity to update data continually for operational use, and the desire to access directly relatively small amounts of data corresponding to selected geographic regions at appropriate times.

The high computer overhead encountered in processing can therefore be minimized by designing an Earth data base structure with constant (but selectable) geometric resolution cells, which are also locally invariant in shape along a translation in any direction. This would eliminate the necessity to account for nonequal-area resolution cells, and also the need to compute the location of every resolution cell in the data base. Moreover, the design should also utilize a fairly simple transformation between the user-preferred geographic coordinates and the internal data base coordinates. This would greatly facilitate arithmetic and transfer operations desired by the user in mathematical computations or in graphic display.

In the previous study (Reference 1), six data base structures were conceptualized. Of these, the most promising was the Quadrilateralized Spherical Cube Data Base. In this model, the sphere is visualized as a spherical cube, as illustrated in Figure 1-1. This spherical cube is obtained by radially projecting the edges of an inscribed cube, as shown in Figure 1-2.

From Figure 1-2, it is obvious that equal-area elements on the plane square do not radially project as equal-area elements on the spherical square. For example, those elements near the center of the plane square have larger projections than those elements near the edges of the plane square. Hence, if a rectangular grid of equal-area elements is first constructed on the plane square, it is then necessary to distort this grid into a curvilinear network so that the elements near the center are smaller than those near the edges. The distortion is such that when the curvilinear elements are projected radially, equal-area elements are again obtained on the spherical square. The desired sequence of transformations is illustrated in Figure 1-3 through 1-5. The mathematical details of deriving these transformations are discussed in Section 2.

For the present, it suffices to say that it is possible to obtain a world map such as Figure 1-6. This map illustrates the continental outlines as they would appear on the cube with the original undistorted rectangular coordinates. This is accomplished by reversing the sequence of transformations previously illustrated by Figures 1-3 through 1-5. Thus, in Figure 1-6, equal-area regions correspond to equal-area regions on the spherical Earth. An examination of this planar equal-area world map shows that the distortion of the continental outlines is not as great as might be expected.

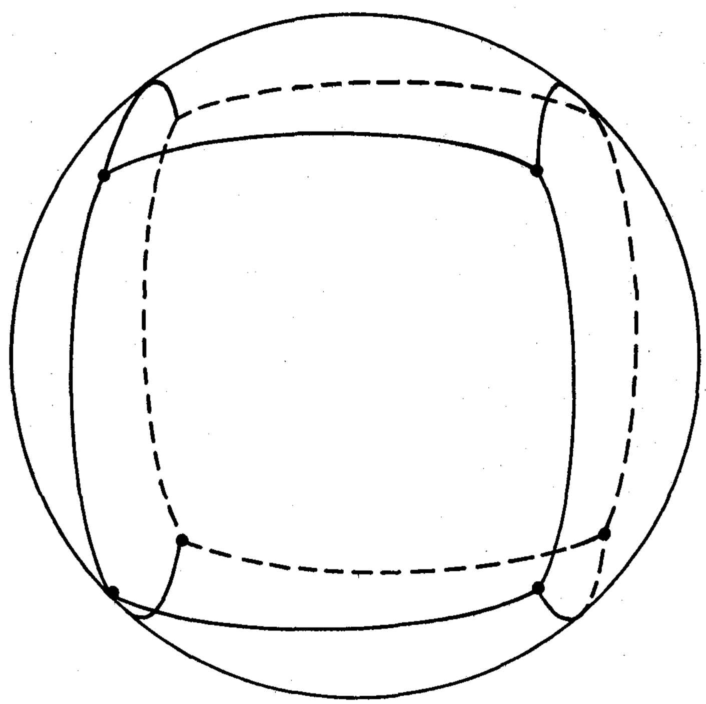

Figure 1-1. Spherical Cube

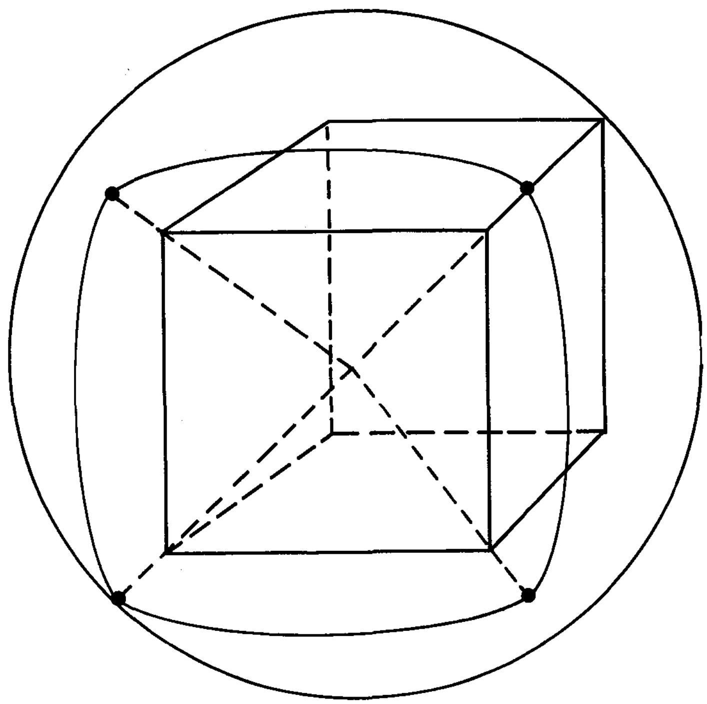

Figure 1-2. Construction of the Spherical Cube

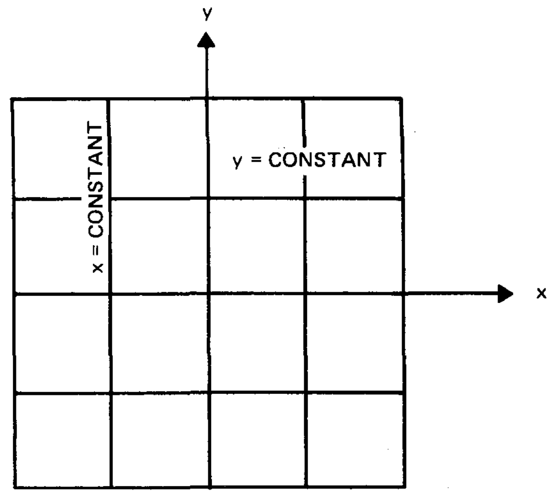

Figure 1-3. Plane Square With Cartesian Coordinates

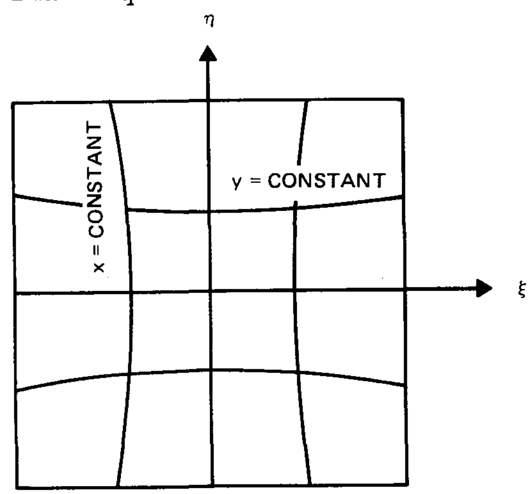

Figure 1-4. Plane Square With Curvilinear Coordinates

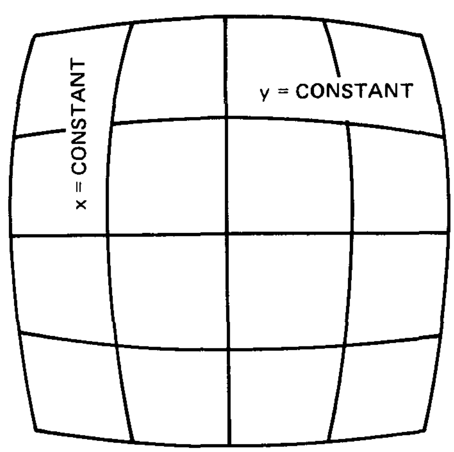

Figure 1-5. Spherical Square With Curvilinear Coordinates

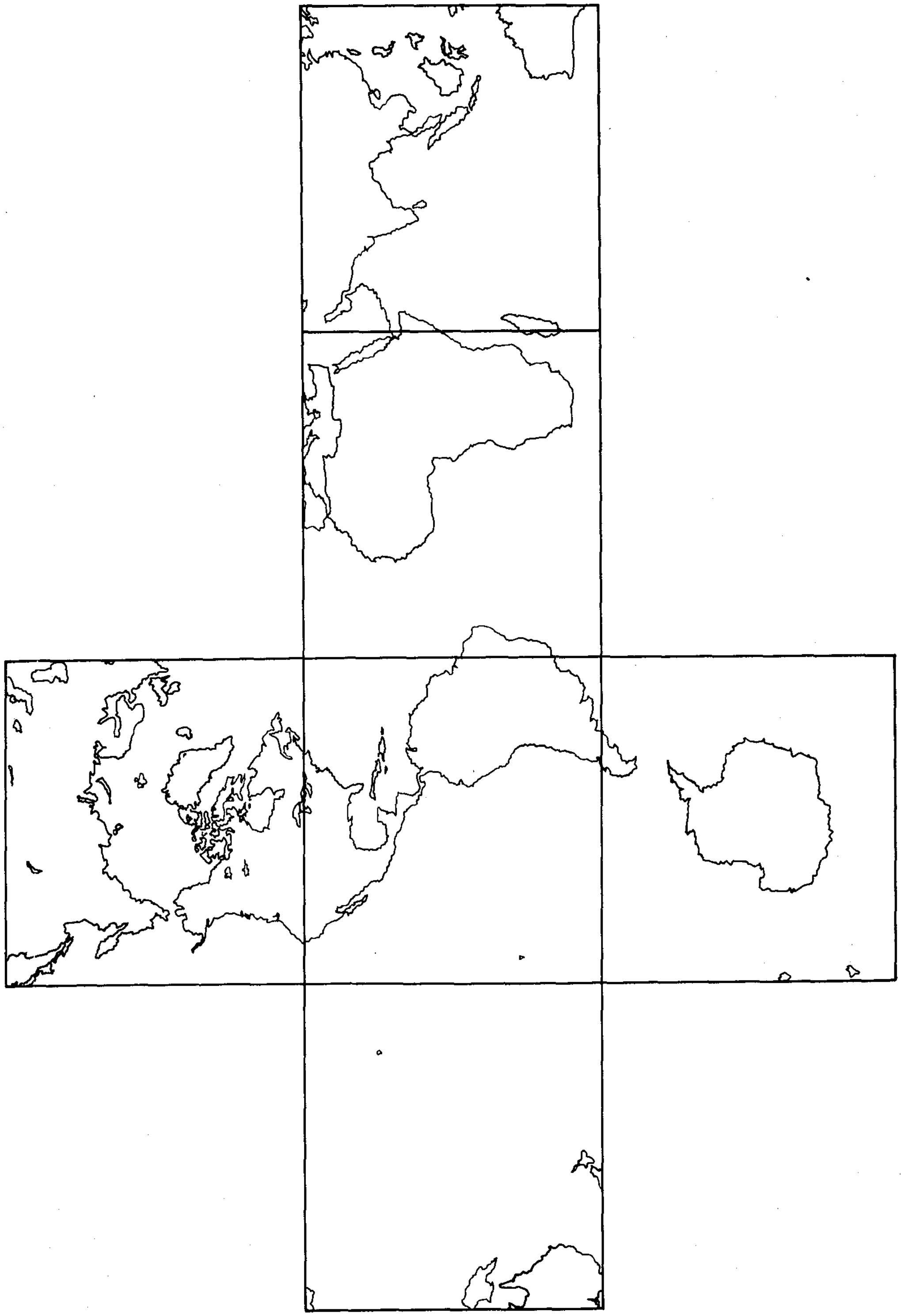

Figure 1-6. World Map on Equal-Area Cube

SECTION 2 - MATHEMATICAL FORMULATION OF DATA BASE STRUCTURE

2.1 SUMMARY

This section is concerned with the derivation of a mapping function which leads to the division of a spherical cube into quadrilaterals of equal area. Specifically, the method involves: (1) dividing a plane square into rectangles of equal area; (2) transforming the rectangular coordinate system on the plane square to a curvilinear coordinate system on the same plane square; and (3) projecting the new nonequal-area elements on the original plane square to a subtending spherical surface such that equal-area spherical surface elements are again obtained. In addition to the direct transformation function described, analysis is also performed for obtaining the inverse transformation function which reverses the above process.

2.2 DERIVATION OF DIRECT MAPPING FUNCTION

First, consider a plane surface subtended by a spherical surface with radius R. Let $\vec{r}_o$ be the vector from the center of the sphere to the given plane. As shown in Figure 2-1, let dA_p be an area element on this plane, and let $\vec{r}$ be the vector from the center of the sphere to the area element dA_p.

Let dA_s be the spherical area element obtained by projecting dA_p radially onto the sphere. Then, it can be readily shown that the following relation between dA_p and dA_s holds:

$$dA_s = \frac{R^2 \cos^3 (\vec{r}, \vec{r}_o)}{r_o^2} \, dA_p \qquad (2\text{-}1)$$

where $(\vec{r}, \vec{r}_o)$ denotes the angle between $\vec{r}$ and $\vec{r}_o$.

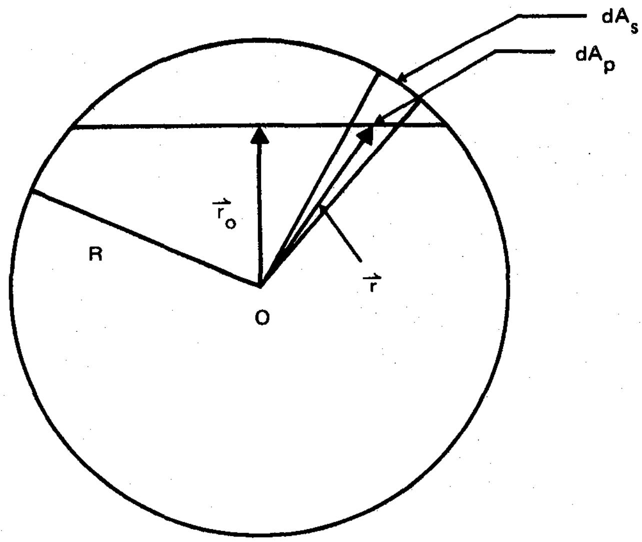

Figure 2-1. Relation Between Plane and Spherical Area Elements

Let $(\xi\ ,\ \eta\ ,\ r_o)$ denote the components of the vector $\vec{r}$. Then, it follows that

$$\cos(\vec{r}, \vec{r}_o) = \frac{r_o}{r} = \frac{r_o}{\left(r_o^2 + \xi^2 + \eta^2\right)^{1/2}} \tag{2-2}$$

Moreover, for convenience, let the unit of length be chosen such that the radius, R, of the sphere is equal to unity. Then, Equations (2-1) and (2-2) yield

$$dA_s = \frac{r_o}{\left(r_o^2 + \xi^2 + \eta^2\right)^{3/2}} dA_p \tag{2-3}$$

Next, consider a cube together with a circumscribing spherical surface. On each of the six plane faces of the cube, a rectangular coordinate system (x, y) may be defined, the domain of definition being $-r_o \leq x\ ,\ y \leq r_o$. It may be easily verified that

$$r_o = \frac{1}{\sqrt{3}} \tag{2-4}$$

Let a new coordinate system $(\xi\ ,\ \eta)$ be defined by

$$\begin{aligned} \xi &= \xi(x, y) \\ \eta &= \eta(x, y) \end{aligned} \tag{2-5}$$

where $\xi(x\ ,\ y)$ and $\eta(x\ ,\ y)$ are independent arbitrary functions.

The new area element $d\xi d\eta$ is related to the original area element dxdy by

$$d\xi d\eta = J\left(\frac{\xi, \eta}{x, y}\right) dxdy \tag{2-6}$$

where $J\left(\frac{\xi, \eta}{x, y}\right)$ is the Jacobian of transformation

$$J\left(\frac{\xi, \eta}{x, y}\right) = \begin{vmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{vmatrix} \tag{2-7}$$

If this new area element is projected radially onto the surface of the sphere, Equations (2-3) and (2-6) yield

$$dA_s = \frac{r_o}{\left(r_o^2 + \xi^2 + \eta^2\right)^{3/2}} J\left(\frac{\xi, \eta}{x, y}\right) dxdy \tag{2-8}$$

which relates the spherical area element dA_s to the original area element dxdy . For original equal-area plane elements dxdy to transform into equal-area spherical elements dA_s , it follows that

$$\frac{r_o}{\left(r_o^2 + \xi^2 + \eta^2\right)^{3/2}} J\left(\frac{\xi, \eta}{x, y}\right) = \lambda^2 \tag{2-9}$$

or

$$\frac{r_o}{\left(r_o^2 + \xi^2 + \eta^2\right)^{3/2}} \left(\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial y} - \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial x}\right) = \lambda^2 \qquad (2\text{-}10)$$

where λ^2 is a constant.

It is easy to verify that the value of λ^2 is equal to the ratio of the area of the spherical square to the area of the plane square, i.e.,

$$\lambda^2 = \frac{2\pi/3}{4\, r_o^2} = \frac{\pi}{2} \qquad (2\text{-}11)$$

An alternative form of Equation (2-10) is

$$\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial y} - \frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial x} = \gamma^2 \left(1 + \frac{\xi^2}{r_o^2} + \frac{\eta^2}{r_o^2}\right)^{3/2} \qquad (2\text{-}12)$$

where

$$\gamma^2 = r_o^2 \lambda^2 = \frac{\pi}{6} \qquad (2\text{-}13)$$

From Equations (2-7) and (2-12), it is seen that γ^2 may be interpreted as the area-scale of transformation at the point $(\xi = 0 \;,\; \eta = 0)$.

Equation (2-12) in itself is quite general. It is now desirable to specify the following general properties for the transformation from (x, y) to (ξ, η).

1. To preserve symmetry in the transformation Equation (2-5), it is required that

$$\xi = f(x, y)$$
$$\eta = f(y, x) \tag{2-14}$$

Equation (2-14) states that ξ and η have exactly the same form of dependence on x and y, except that the roles of x and y are interchanged. Moreover, symmetry preservation also requires that the function $f(x, y)$ be odd in x and even in y, i.e.,

$$f(-x, y) = -f(x, y)$$
$$f(x, -y) = f(x, y) \tag{2-15}$$

As a consequence of Equations (2-14) and (2-15), it is seen that the origin maps back into itself, i.e.,

$$f(0, y) = 0 \tag{2-16}$$

2. To map points on the sides of the square back into points on the same sides, it is necessary that

$$f(r_o, y) = r_o \tag{2-17}$$

As a consequence of all the above requirements, it may be shown that $\frac{\partial \xi}{\partial y}$ and $\frac{\partial \eta}{\partial x}$ are zero at the points $(0\ ,\ 0)$ and $(r_o\ ,\ r_o)$. Therefore, from Equation (2-12), it follows that

$$\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0\\ y=0}} = \left.\frac{\partial \eta}{\partial y}\right|_{\substack{x=0\\ y=0}} = \gamma = \sqrt{\frac{\pi}{6}} = 0.72360\ \ 12545\ \ 582 \tag{2-18}$$

$$\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o\\ y=r_o}} = \left.\frac{\partial \eta}{\partial y}\right|_{\substack{x=r_o\\ y=r_o}} = \mu = \sqrt{\frac{\sqrt{3}\pi}{2}} = 1.6494\ \ 54166\ \ 187 \tag{2-19}$$

If $f(x\ ,\ y)$ can be expanded in a power series in x and y, then Equation (2-16) requires that

$$f(x,\ y) = x \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij}\, x^{2i}\, y^{2j} \tag{2-20}$$

The condition in Equation (2-18) yields

$$a_{00} = \gamma \tag{2-21}$$

The condition in Equation (2-17) may be incorporated into $f(x\ ,\ y)$ by writing it in the form

$$f(x,\ y) = \gamma x + \frac{(1-\gamma)}{r_o^2} x^3 + \left(r_o^2 - x^2\right) x \sum_{(i+j)\geq 1}^{\infty} b_{ij}\, x^{2i}\, y^{2j} \tag{2-22}$$

It may be shown that Equation (2-12), together with the conditions given by Equations (2-16) through (2-18), are not sufficient to determine uniquely the transformation in Equation (2-14). This nonuniqueness is manifested by the fact that there are more unknowns (b_{ij}) than equations when Equation (2-22) is substituted into Equation (2-12) and terms of the same degree are equated.

Finally, to incorporate the condition in Equation (2-19), it is most efficient to express $f(x, y)$ in the following form. The details for arriving at this form are given in Appendix A.

$$f(x, y) = \gamma x + \frac{(1-\gamma)}{r_o^2} x^3 + xy^2 \left(r_o^2 - x^2\right) \left[\delta + \left(r_o^2 - y^2\right) \sum_{\substack{i \geq o \\ j \geq o}} c_{ij} x^{2i} y^{2j}\right] + x^3 \left(r_o^2 - x^2\right) \left[\omega + \left(r_o^2 - x^2\right) \sum_{i \geq o} d_i x^{2i}\right] \tag{2-23}$$

where

$$\delta = \frac{1}{4r_o^4} \left[-(\mu + 2\gamma) + \sqrt{(\mu^2 - 4\mu\gamma + 4\gamma^2 + 16\sqrt{2}\,\gamma^2)}\right] \tag{2-24}$$

$$= 0.79048\ 64491\ 208$$

$$\omega = \frac{1}{2r_o^4} \left(3 - 2\gamma - \mu - 2r_o^4 \delta\right) \tag{2-25}$$

$$= -\ 1.2254\ 41487\ 984$$

An approximate mapping function may be obtained by truncating the series expansion in Equation (2-23) at some degree, and then obtaining the coefficients c_{ij} and d_i which minimize the following residual function:

$$\phi(c_{ij}, d_i) \equiv \int_{-r_o}^{r_o} \int_{-r_o}^{r_o} \left[r_o^{-2} \left(1 + \frac{\xi^2}{r_o^2} + \frac{\eta^2}{r_o^2} \right)^{-3/2} \times \left(\frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial y} - \frac{\partial \xi}{\partial y} \frac{\partial \eta}{\partial x} \right) - \lambda^2 \right]^2 dxdy \tag{2-26}$$

This residual function is obtained by considering Equation (2-10) or (2-12). Then, $\phi(c_{ij}, d_i)$ is evidently equal to zero for the exact transformation function $f(x, y)$. For computational purposes, Equation (2-26) is replaced by

$$\phi(c_{ij}, d_i) = \sum_{x_k} \sum_{y_l} \left[r_o^{-2} \left(1 + \frac{\xi^2}{r_o^2} + \frac{\eta^2}{r_o^2} \right)^{-3/2} \times \left(\frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial y} - \frac{\partial \xi}{\partial y} \frac{\partial \eta}{\partial x} \right) - \lambda^2 \right]^2 \tag{2-27}$$

where the points (x_k, y_l) are chosen to form a regular grid over the plane square. A computer software program for performing this minimization problem is given in Appendix E. For a second-degree approximation of the series in Equation (2-23), the following values of c_{ij} and d_i are obtained:

$$c_{00} = -2.7217\ 05366\ 1814$$

$$c_{10} = -5.5842\ 16830\ 5430$$

$$c_{01} = 2.1711\ 17480\ 9423$$

$$c_{20} = -3.4578\ 62747\ 3390$$

$$c_{11} = -6.4160\ 15152\ 6783$$

$$c_{02} = 1.9736\ 26575\ 8872$$

$$d_0 = 1.4833\ 12929\ 4187$$

$$d_1 = 1.1199\ 72606\ 9742$$

$$d_2 = 6.0515\ 38216\ 1464$$

The corresponding mapping function $f(x, y)$ is accurate to about five significant figures.

2.3 DERIVATION OF INVERSE MAPPING FUNCTION

Corresponding to the symmetrical direct mapping function expressed in Equation (2-14), it may be verified that the inverse mapping function is also symmetrical, i.e.,

$$\begin{aligned} x &= f^*(\xi, \eta) \\ y &= f^*(\eta, \xi) \end{aligned} \qquad (2\text{-}28)$$

As discussed in Appendix A, $f^*(\xi, \eta)$ must be expressed in the form

$$f^*(\xi, \eta) = \gamma^*\xi + \frac{(1-\gamma^*)}{r_o^2}\xi^3 + \xi\eta^2\left(r_o^2 - \xi^2\right)\left[\delta^* + \delta_1^*\left(r_o^2 - \xi^2\right) + \left(r_o^2 - \eta^2\right)\sum_{\substack{i\geq 0 \\ j\geq 0}} c_{ij}^*\,\xi^{2i}\eta^{2j}\right] + \xi^3\left(r_o^2 - \xi^2\right)\left[\omega^* + \left(r_o^2 - \xi^2\right)\sum_{i\geq 0} d_i^*\,\xi^{2i}\right] \tag{2-29}$$

where

$$\begin{aligned}
\gamma^* &= \frac{1}{\gamma} \\
\mu^* &= \frac{1}{\mu} \\
\gamma_1^* &= \frac{1}{\gamma + r_o^4\,\delta} \\
\mu_1^* &= \frac{1}{\mu + 2r_o^4\,\delta} \\
\delta^* &= \frac{(\mu_1^* - \mu^*)}{2r_o^4} \\
\omega^* &= \frac{1}{2r_o^4}\left(3 - 2\gamma^* - \mu^* - 2r_o^4\,\delta^*\right) \\
\delta_1^* &= \frac{1}{r_o^2}\left[\frac{(\gamma_1^* - \gamma^*)}{r_o^4} - \delta^*\right]
\end{aligned} \tag{2-30}$$

An approximate inverse mapping function may be obtained by truncating the series expansion in Equation (2-29) at some degree, and then obtaining the coefficients c_{ij}^* and d_i^* which minimize the following residual function:

$$\phi^*(c_{ij}^*,\ d_i^*) \equiv \int_{-r_o}^{r_o} \int_{-r_o}^{r_o} \left\{ \left[x - f^* \left(f(x,y),\ f(y,x) \right) \right]^2 \right. \qquad (2\text{-}31)$$

$$\left. + \left[y - f^* \left(f(y,x),\ f(x,y) \right) \right]^2 \right\}^{1/2} dx\, dy$$

In obtaining this residual function, the direct mapping function $f(x\ ,\ y)$ is considered, as given by Equation (2-23). Then, $\phi^*(c_{ij}^*\ ,\ d_i^*)$ is evidently equal to zero for the exact inverse mapping function $f^*(\xi\ ,\ \eta)$. Again, for computational purposes, Equation (2-31) is replace by

$$\phi^*(c_{ij}^*,\ d_i^*) = \sum_{x_k} \sum_{y_\ell} \left\{ \left[x - f^* \left(f(x,y),\ f(y,x) \right) \right]^2 \right. \qquad (2\text{-}32)$$

$$\left. + \left[y - f^* \left(f(y,x),\ f(x,y) \right) \right]^2 \right\}^{1/2}$$

A computer software program for performing this minimization problem is also given in Appendix E. For a second-degree approximation of the series in Equation (2-29), the following values of c_{ij}^* and d_i^* were obtained:

$$c_{00}^* = 3.973\ \ 89249$$

$$c_{10}^* = 6.591\ \ 19476$$

$$c_{01}^* = -25.368\ \ 92536$$

$$c_{20}^* = -73.064\ \ 97000$$

$$c_{11}^* = 77.381\ 61133$$

$$c_{02}^* = 21.685\ 89623$$

$$d_0^* = 1.811\ 28250$$

$$d_1^* = 37.635\ 47857$$

$$d_2^* = 63.000\ 23655$$

The corresponding mapping function $f^*(\xi\ ,\ \eta)$ is accurate to about five significant figures.

SECTION 3 - ORGANIZATION OF DATA BASE

3.1 SUMMARY

This section mainly is concerned with the organization of the data base, whose basic structure is derived from the equal-area division of the spherical cube. It suffices, therefore, to limit most of the discussion to just one spherical square. The binary indexing scheme of the data base will be described. The reasons for adopting this indexing scheme are then discussed. This is then followed by an outline for computing the serial string index for the cell addresses.

3.2 INDEXING OF DATA BASE

The nature of the data base system adopted is extremely simple since all data are stored in a single large array. This organization is natural since:

1. The contemplated data base corresponds to a completely filled array
2. No data item has other than a positional relationship to another

Thus, all data in the data base may be accessed by directly calculated positional indices with no storage inefficiencies due to unused areas of the array.

The mapping of the data base onto storage may be accomplished in a number of ways. The classic method is mapping by columns and rows, as is done in FORTRAN array storage and as implied in the usual mathematical matrix notation. Other mapping techniques include scatter storage in which a psuedo-random, one-to-one mapping is employed as well as various consistent logical stringing techniques.

The specific scheme studied in this report is based on the binary division scheme discussed in Reference 1. This mapping starts at the level of the faces

of the spherical cube, numbering these faces 1 through 6 as in Figure 3-1. Each face is divided, to the requisite resolution level, by a two-dimensional binary grid, as shown in Figure 3-2. On each level of division, the areas are divided into quadrants, which are labeled by a 2-bit binary number. Each level of division, n , is indicated by the addition of two binary bits to the least-significant end of a 2N bit binary number. This binary index is the serial location of a point in the 2N location data array.

An address to the third level of division would appear as in Figure 3-3.

This organization will be referred to as the serial addressing scheme, and the data array as the serial data string.

3.3 REASONS FOR ADOPTING INDEXING SCHEME

In developing a method of ordering data in the data base, there are three obvious advantages to using a serial addressing scheme rather than a normal column and row addressing scheme:

1. Reduction in I/O time through maintenance of near-neighbor relationships
2. Compactness of arrays containing addresses
3. Maintenance of a consistent addressing scheme regardless of resolution level

The serial addressing scheme reduces I/O time for disk type storage devices because more near neighbors of a point are within the range which requires no arm motion for accessing. A specific example is provided in Section 3.5. The expression of addresses as a single bit string allows storage of addresses as single machine words, whereas a two-dimensional addressing scheme would require two or three words, including one for the face number. Finally, the expandibility and generality of the serial string permits the use at any resolution

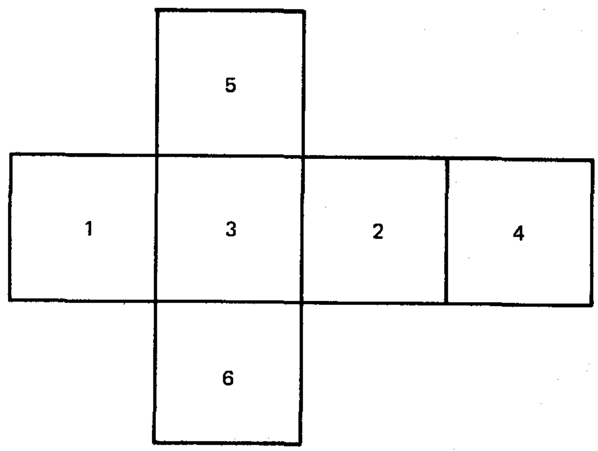

Figure 3-1. Face Numbering Scheme

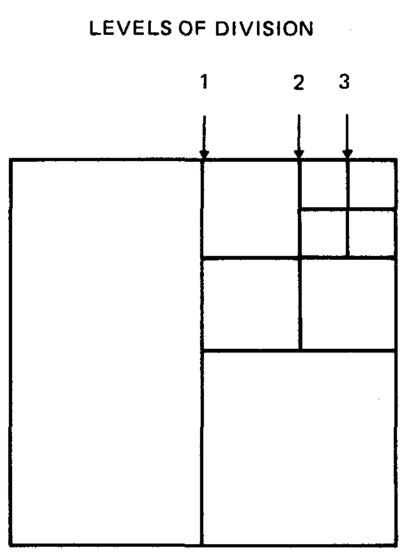

Figure 3-2. Binary Division Scheme

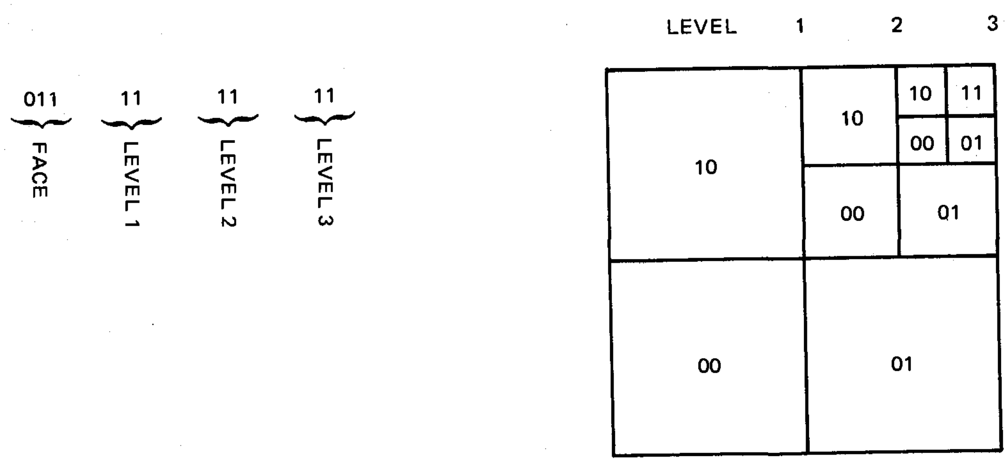

Figure 3-3. Illustrative Labeling by Binary Bits

level without regard to physical storage considerations, such as record size. Any reasonable matrix type storage scheme would require a dual (or multiple) level of addresses for record and item within record location in the serial scheme. This is accomplished simply by considering the high order m bits as the record number, and the low order $n - m$ bits as the address within record.

A major advantage discovered while studying implementation of this scheme was the extreme rapidity of index calculation. This last advantage is gained through changing the basic linking from the previously proposed pattern, shown in Figure 3-4, to that shown in Figure 3-5. This pattern retains all of the advantages of the former scheme, and also permits a simpler calculation of the serial index.

The motivation for this pattern may be best understood by considering the data base as a space filling binary tree structure in two dimensions. Figure 3-6 illustrates a simple one-dimensional tree and an associated binary numbering scheme. Figure 3-7 extends this system to two dimensions, from which it is seen that it is possible to combine the two coordinates into a one-dimensional index. This is accomplished by utilizing the even bits of a binary number to represent one coordinate and the odd bits to represent the other. Table 3-1 illustrates the values assumed by each coordinate.

For example, the addresses of the cells (1, 1) and (3, 5) will be represented in binary notation as follows:

$$(1, 1) = 01 + 10 = 11$$

$$(3, 5) = 101 + 100010 = 100111$$

The x and y coordinates may be recovered from the index by simply masking out the even or odd bits.

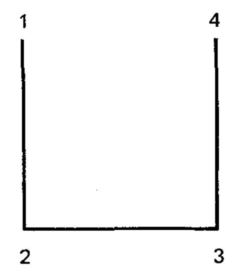

Figure 3-4. Previously Proposed Stringing Pattern

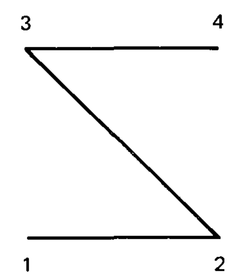

Figure 3-5. Presently Proposed Stringing Pattern

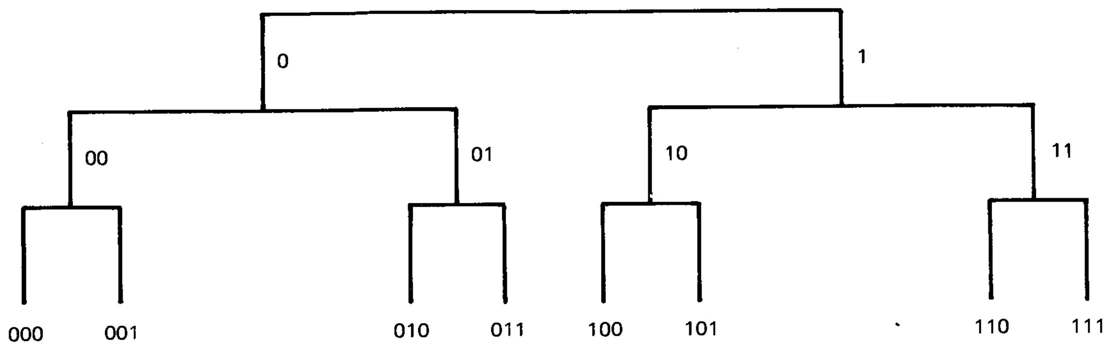

Figure 3-6. One-Dimensional Binary Tree

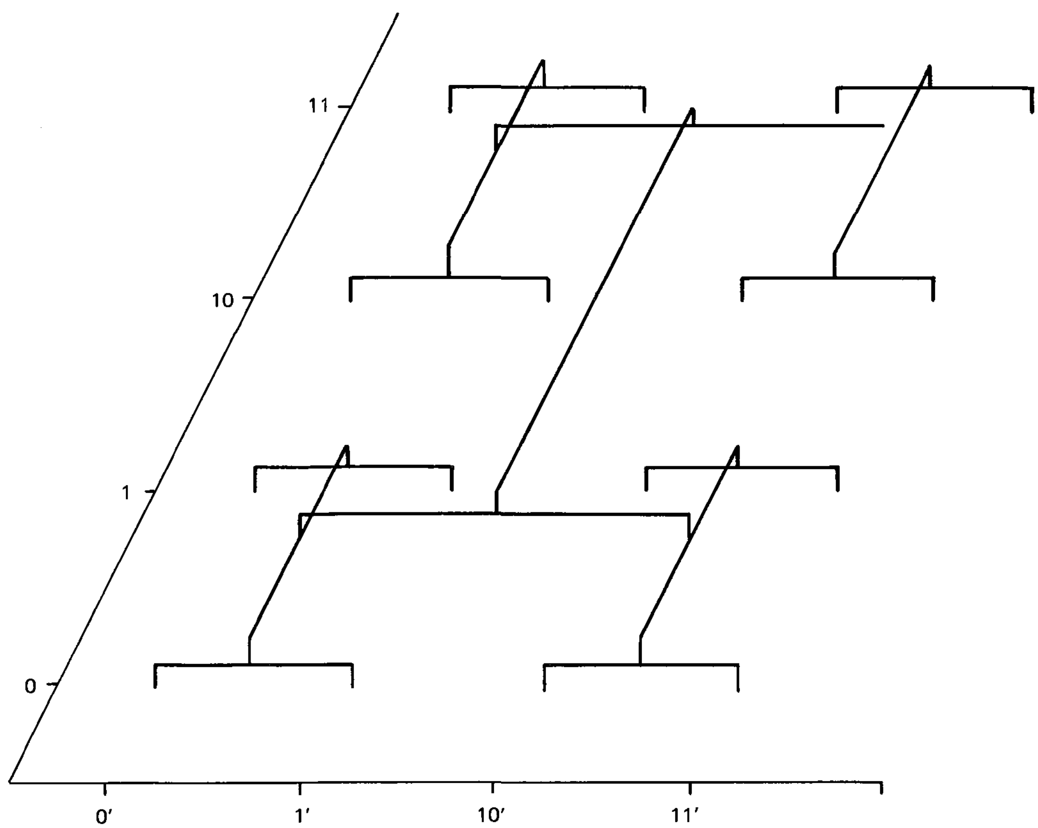

Figure 3-7. Two-Dimensional Binary Tree

Table 3-1. Binary Representation of Coordinates

DECIMAL VALUE	BINARY	X-COORDINATE (ODD)	Y-COORDINATE (EVEN)
1	1	01	10
2	10	100	1000
3	11	101	1010
4	100	10000	100000
5	101	10001	100010
6	110	10100	101000
7	111	10101	101010

In summary, the binary indexing scheme and the bit-string ordering for the first two levels of division are shown in Figure 3-8 and 3-9.

This linkage yields the desired serial string at any level. It also allows a computation of the serial address which requires only two table look-ups and an add operation.

3.4 CALCULATION OF SERIAL STRING INDEX

As mentioned in the previous section, the serial index is the composite of the x and y indices. The most rapid way of obtaining these indices is by table look-up. The construction of the y (even bit) table may be done as follows:

1. Looping over all bits in the table index

2. Examining the index bit by bit (when bit n in the index is on, set the bit 2n in the table entry)

The x (odd bit) table may be obtained from the even bit table by right-shifting the even bit table entries (Table 3-1).

The calculation of the serial string index may also be accomplished by the construction of a very simple hardware device. This device would consist of three registers: an x register, a y register, and a serial string or s register.

Two register-to-register instructions would provide packing from x , y to s and unpacking s to x , y . These instructions would initiate parallel transfer from the two n-bit coordinate registers to the 2n-bit serial register and vice versa. The interconnection is shown in Figure 3-10.

3.5 NEAR-NEIGHBOR DISTANCES IN SERIAL STRING

The serial stringing described in Section 3.3 leads to a reduction in disk I/O time because of a reduction of the number of seek operations. This happens

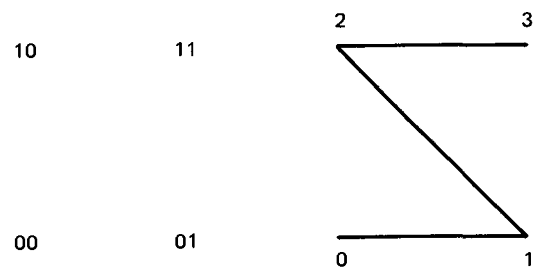

Figure 3-8. First Level of Division

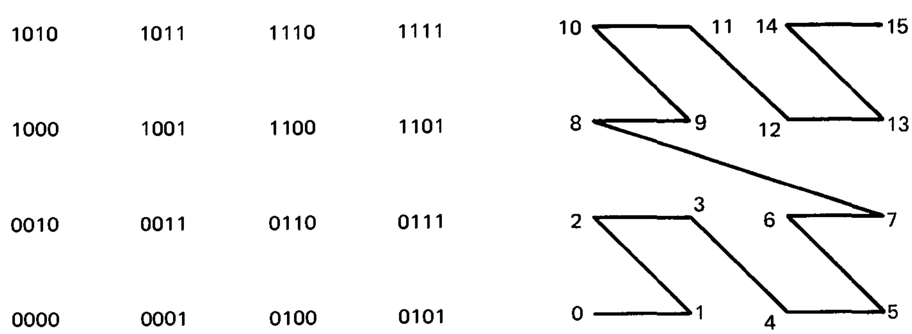

Figure 3-9. Second Level of Division

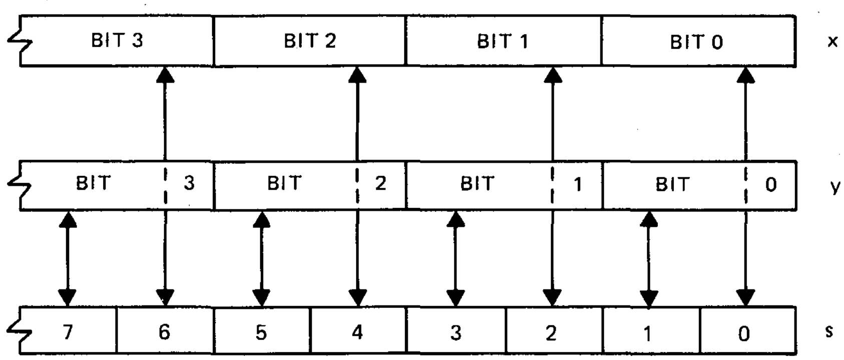

Figure 3-10. Transfer Between Registers

because the number of large gaps separating typical adjoining records is fewer for the serial string than for a matrix type storage scheme.

Figure 3-11 illustrates a resolution element and its eight adjoining spatial neighbors. The circle of radius, $r = a$, is the locus of points one resolution element away from the current point. For this situation, the probabilities of the neighbor being horizontally, vertically, or diagonally adjacent are

$$P_{\bar{h}} = 1/3$$

$$P_{\bar{v}} = 1/3$$

$$P_{\bar{d}} = 1/3$$

The individual probabilities for a specific neighbor are

$$P_h = 1/6$$

$$P_v = 1/6$$

$$P_d = 1/6$$

The matrix storage scheme (column and row storage) maintains a near-neighbor relationship only along the horizontal or vertical axis. Figure 3-12 is a graph of the probability of finding a near neighbor at a given serial separation in an $n \times n$ matrix arranged in columns and rows. The same graph for the serial string described in Section 3.3 (for a 64×64 array) appears in Figure 3-13. Depending on the definition of nearness in the serial string and the size of the array, a gain of 50 percent to 100 percent can be realized by using the serial string scheme.

The case of multi-element record stringing differs somewhat from the simple resolution elements relationship described above. In propagating the

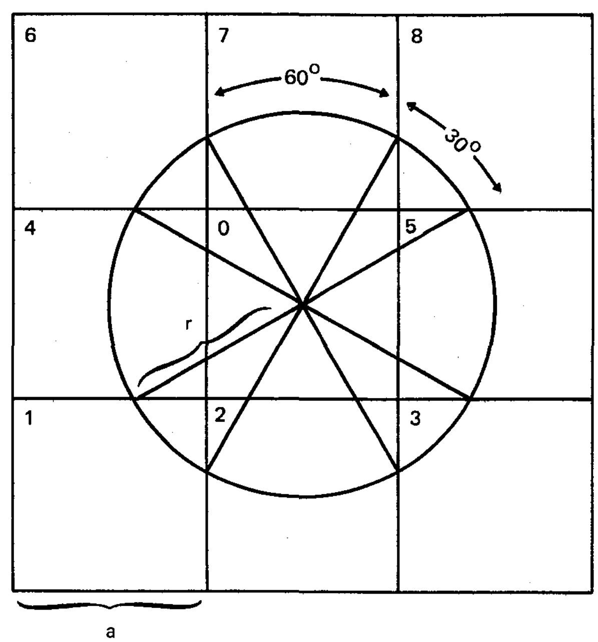

Figure 3-11. Resolution Element and Near Neighbors

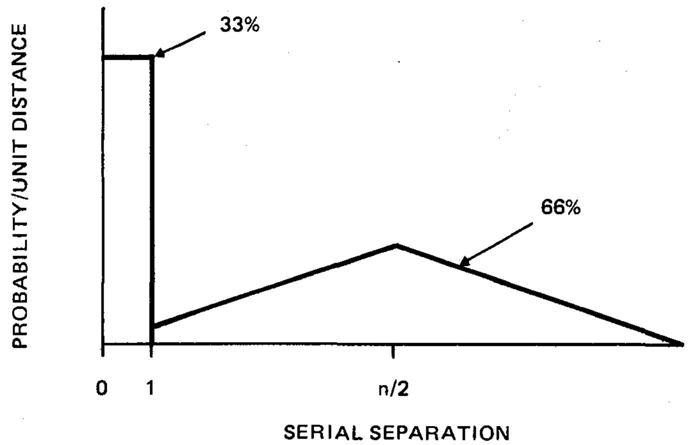

Figure 3-12. Separation of Near Neighbors in Matrix Storage

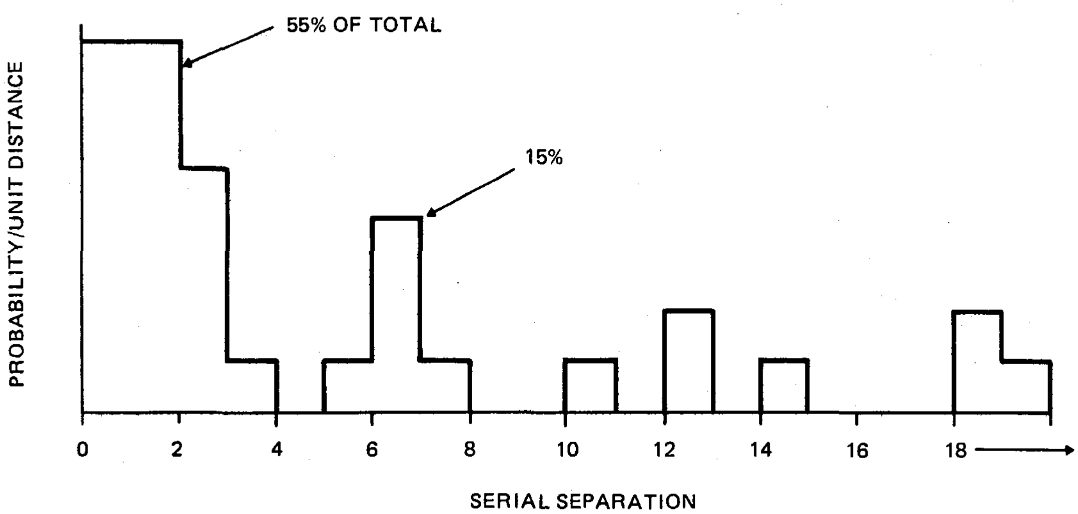

Figure 3-13. Separation of Near Neighbors in Serial String

interpolation vector, r is much less than a, and P_d becomes very small. $P_{\bar{h}}$ and $P_{\bar{v}}$ remain equal and are approximately equal to 1/2. Taking ±8 records as a reasonable cylinder size, an advantage of 7/5 is found for the serial scheme.

SECTION 4 - DATA STORAGE

4.1 SUMMARY

This section is concerned with storing data into the data base by the method of "fast-filling." The method involves the determination of benchmark points corresponding to specific scan resolution elements (pixels). The filling of neighboring data base points corresponding to neighboring pixels is then achieved by interpolation, using appropriate combinations of data base vectors and scan element vectors. This method of data base filling is possible because of the local translational invariance of the division of the spherical cube described in Section 2.

4.2 CONVERSION FROM GEOGRAPHIC COORDINATES TO DATA BASE COORDINATES

The basic objective of the fast-filling method is the insertion of data into the data base with the minimum of computation. For this purpose, it is assumed that over some small region the mapping of scan lines onto the data base yields nearly straight lines, and that the location of points along these lines can be found by interpolation. It is also assumed that there is no editing of data base information. Rather, all data are inserted into the data base as though it were initially empty.

For defining the ground location of a given pixel, the following three basic sets of information are necessary:

- Satellite orbital position
- Satellite attitude
- Time of scan

The determination of ground location (or geographic coordinates) as a function of the above parameters is described in Appendix B. Once the geographic coordinates are available, it is straightforward to obtain the corresponding set of

Cartesian coordinates (a , b , c) in the three-dimensional space. The subsequent mapping of this point into a point onto the cube representing the data base requires the following four operations:

1. Determination of the face onto which the point is to be mapped

2. Projection from the Cartesian (a , b , c) system into the curvilinear (ξ, η) system on a plane square

3. Transformation from (ξ, η) into Cartesian (x , y) coordinates on a plane square

4. Conversion from face number and (x , y) to serial location in the data base

A simple method of face determination arises naturally when a suitable relationship is chosen between the Cartesian system and the data base coordinate system. This system, illustrated in Figure 4-1, is one where each face of the data base cube is perpendicular to a major axis of the Cartesian system whose origin is at the center of the cube.

The face to which a point (a , b , c) belongs is determined by finding the component with the largest absolute value. This point is then projected into the face containing the axis corresponding to the largest component and to its sign. It should be noted at this point, however, that a point can be projected onto any of three planes, two of which are extensions of other faces. This projection is illustrated in Figure 4-2. These projections onto extended faces will have application later.

The projection from the (a , b , c) system onto the face of the cube requires the adoption of a consistent set of face coordinates. The set of face coordinate and face numbering used in Section 3 and their relation to the Cartesian system is illustrated in Figure 4-3. The projection is illustrated in Figure 4-4.

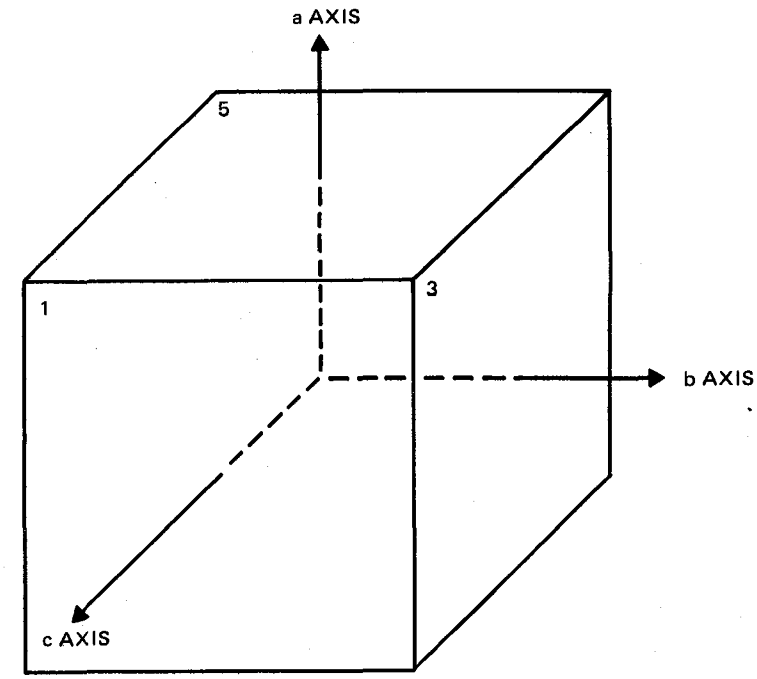

Figure 4-1. Relationship of Cartesian and Data Base Coordinate System

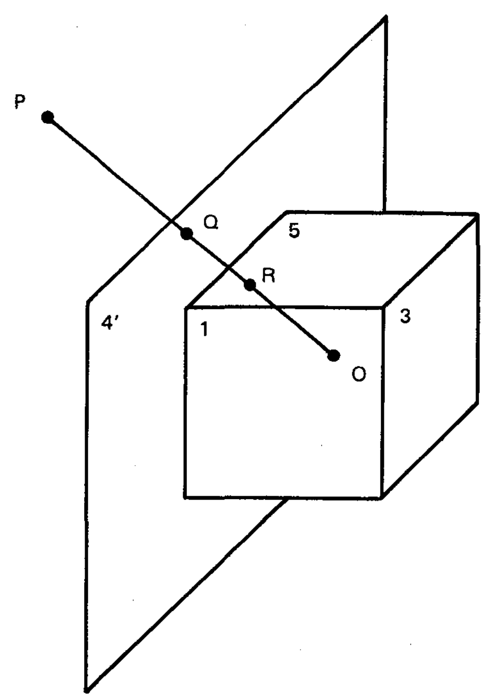

Figure 4-2. Projection of Point P Onto Face 5(R) and the Extension of Face 4(Q)

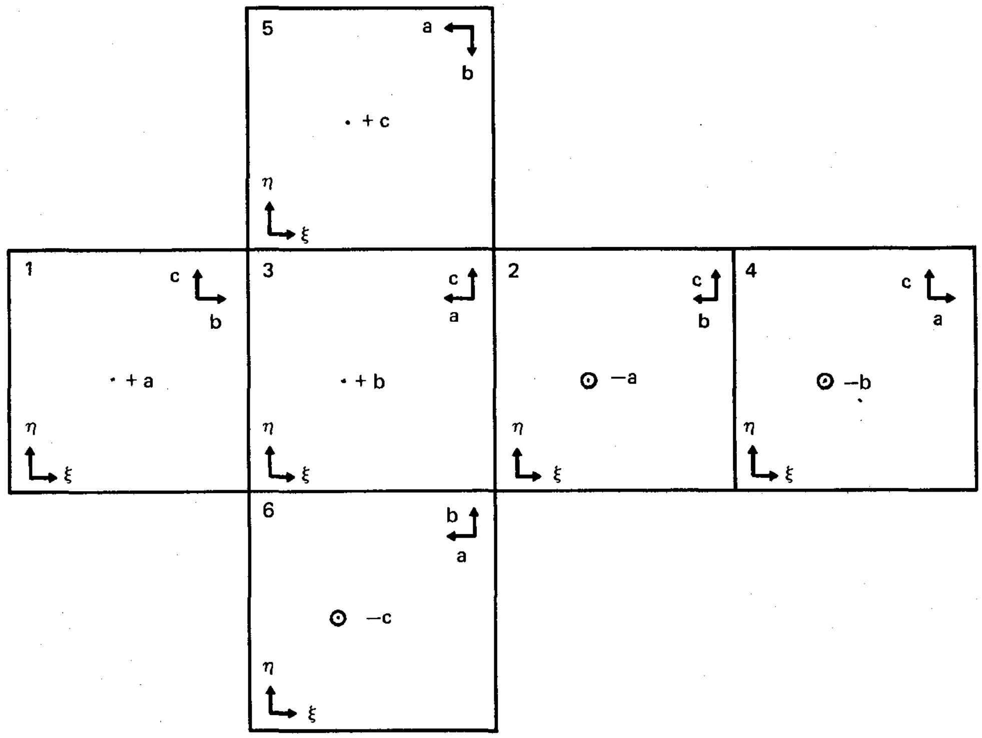

Figure 4-3. Relation of Cartesian and Face Coordinate Systems

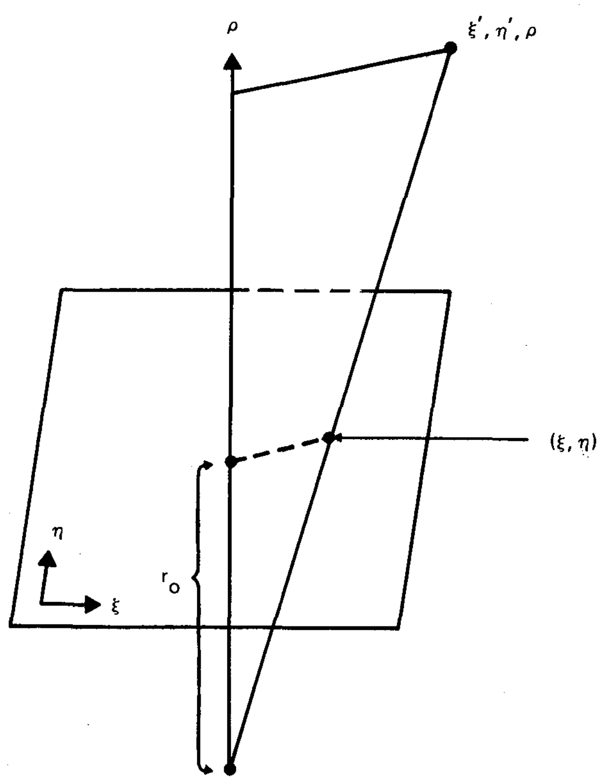

Figure 4-4. Projection of Point (ξ', η', ρ) Onto a Face

From the geometry in Figure 4-4, it is seen that the cube centered Cartesian components of the point (ξ', η', ρ) project onto the cube face (ξ, η) as follows:

$$\begin{aligned} \xi &= \xi' P \\ \eta &= \eta' P \end{aligned} \tag{4-1}$$

where $P = r_o/\rho$

From Figure 4-3, Table 4-1 is constructed illustrating relations between the (a, b, c) and (ξ', η', ρ) systems.

Table 4-1. Relations Between (a, b, c) and (ξ', η', ρ) Systems

FACE NUMBER	ξ'	η'	ρ
1	b	c	a
2	−b	c	−a
3	−a	c	b
4	a	c	−b
5	−a	−b	c
6	−a	b	−c

Using these relations, any point (a, b, c) may be mapped into an appropriate (ξ, η) system.

The transformation from the curvilinear (ξ , η) system to the rectangular data base coordinates (x , y) has been described in Section 2. The relevant equations are

$$
\begin{aligned}
x &= f^*(\xi , \eta) \\
y &= f^*(\eta , \xi)
\end{aligned}
\qquad (4-2)
$$

The serial data base location is obtained by table look-up as described in Section 3 or by direct calculation.

Clearly, the application of this mapping procedure to every data point would be inefficient. Consequently, it is desirable to seek methods for reducing the number of computations.

4.3 METHOD OF INTERPOLATION

The simplest method for reducing the number of computations is to utilize an interpolation method. This interpolation may be done at any one of a number of stages of the mapping from scanner coordinates to data base coordinates. The greatest reduction in computation is achieved by performing, in the data base coordinate system, a linear interpolation both along the scan line and across scan lines. Neglecting edge effects, interpolation in this frame reduces the number of transformation by a factor $N \times M$, where N is the number of pixels between calculated points, and M is the number of scan lines between calculated points. These points will be referred to as benchmarks, and the grid they form as the benchmark grid.

Interpolation (Figure 4-5) is accomplished by taking three points on the benchmark grid and constructing displacement vectors in the along-scan and across-scan directions.

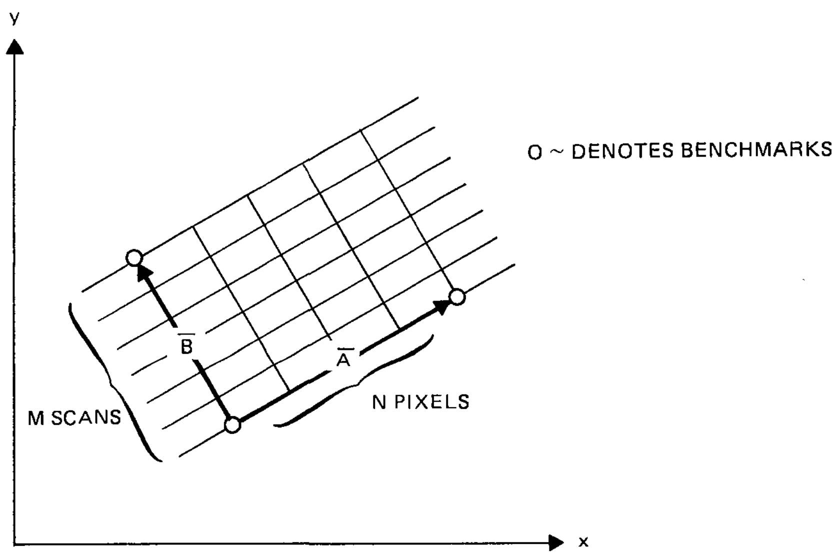

Figure 4-5. Interpolation Using Displacement Vectors

The unit displacement vectors $\vec{a}$, $\vec{b}$ are obtained from

$$\vec{a} = \vec{A}/N$$
$$\vec{b} = \vec{B}/M \tag{4-3}$$

Consider a pixel at point P offset by n pixels and m scan lines from the benchmark O . Then its coordinates are given by

$$\vec{OP} = n\vec{a} + m\vec{b} \tag{4-4}$$

Two factors affecting the range of interpolation are

- Curvature of scan lines in the (x , y) system
- Parallelism of scan lines over the interpolation range

The first effect was investigated by constructing great circle arcs of varying length and orientation, mapping points on these arcs into the data base, and finally comparing the mapped positions with positions obtained by interpolation between the ends of the arc. Interpolated positions were required to map onto calculated positions to an accuracy of 0.25 array element. This procedure was carried out for sets of arcs with origins at the intersections of a 5 x 5 grid covering one quadrant. Table 4-2 presents the results of this procedure.

Non-parallelism of scan lines would restrict the number of scan lines over which interpolation ranges. The primary contributor to non-parallelism would be the use of the return-scan data in an oscillating mirror scanner system. This problem does not exist for coarse resolution data since coarse resolution is averaged over a number of scans. For fine resolution, this problem can be eliminated by separating the scan lines into odd and even scans and interpolating separately.

Table 4-2. Maximum Interpolation Range as a Function of Position in x , y Array of Dimension 4096×4096 (Maximum Error 0.25 Element)

ELEMENT	CENTER	0		1024		2048	
	0	1024	1024	512	256	64	0
		1024	1024	512	256	64	
	1024	512	512	256	128	32	1024
		256	256	128	64	32	
	2048	64	64	32	32	16	2048
		0		1024		2048	VERTEX

ELEMENT

To maximize computational efficiency, it is desirable to interpolate over the longest range possible. Two schemes for determining this range present themselves:

- Store precalculated range in a table
- Attempt interpolation to a bench point and reduce range if interpolation is inaccurate

The precalculation method has an advantage for a system of constant resolution. The attempted interpolation method would be simpler to implement and is also applicable for variable resolution.

The largest feasible interpolation region makes a sensible logical block over which to perform computations. Such $N \times M$ rectangular blocks of data will occasionally cross a face boundary. Clearly, interpolation between points in differing (x , y) systems is meaningless. Thus a scheme for crossing these boundary regions is necessary.

A simple logical check will identify blocks which cross face boundaries. The face number bits in the four-corner benchmarks serial position can be checked against each other. If all four corners lie in one face, all points in the block must lie in that face; if any corner lies on a differing face, special measures must be taken.

Assuming that the interpolation blocks are small compared to a face, two types of face crossing may occur. These are edge crossing and vertex crossing, as illustrated in Figure 4-6.

Two types of vertex crossing may occur (Figure 4-7). Type 1 is characterized by a block having corners on three faces, and this situation may be distinguished immediately. Type 2 is characterized by a block having corners on only two faces and is difficult to distinguish from a simple edge crossing. To determine

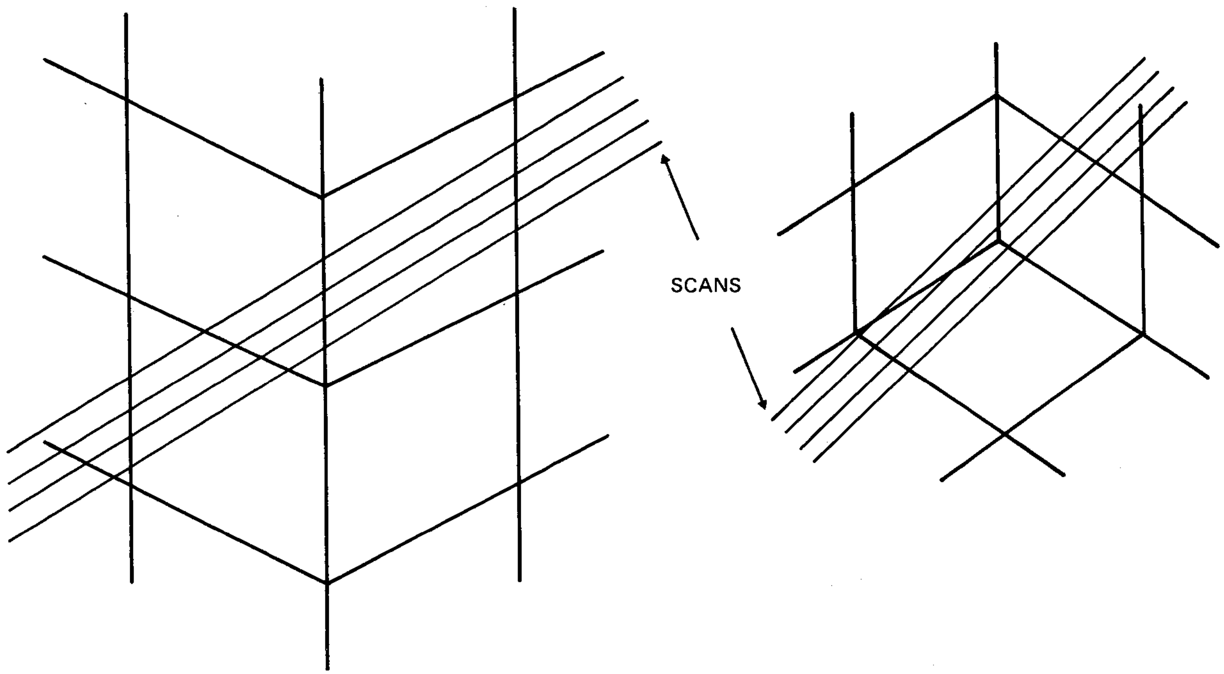

Figure 4-6. Edge Crossing and Vertex Crossing

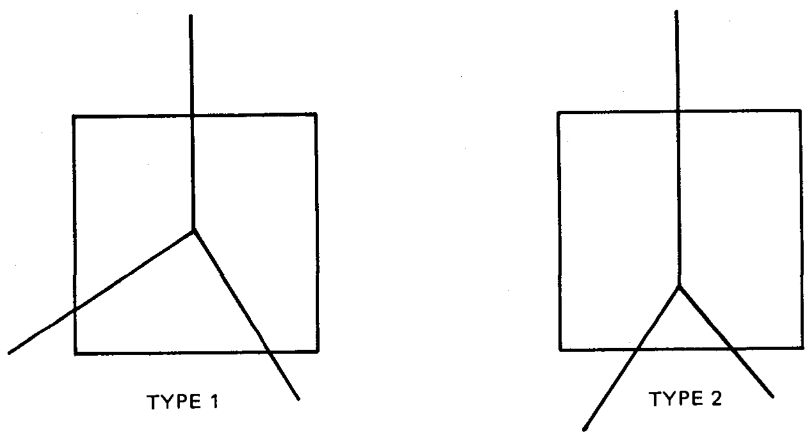

Figure 4-7. Two Types of Vertex Crossing

whether a block contains an edge crossing or a type 2 vertex crossing, the following procedure may be used:

1. Choose one face coordinate system and calculate the position of all four corners in this coordinate system. (Some of these points will lie on the extension of the face.)

2. Test this quadrilateral to see if it contains a vertex. Two tests may be used:

   a. Construct the lines connecting the four corner benchmarks and, proceeding around the quadrilateral, determine whether or not the vertex lies to the same side of the lines connecting the benchmarks. If it does, the vertex is within the block.

   b. A more elegant scheme is based on the fact that all of the interpolation blocks will be very nearly rhomboidal. Given this, it is then determined whether a point lies within or without the rhombus by comparing the areas of the triangles formed when the vertex point is joined to two points on the rhombus. If any of the four possible triangles has an area greater than half the rhombus area, the vertex lies outside as illustrated in Figure 4-8.

When it has been determined which and how many faces of the data base cube are involved, the interpolation method then proceeds to calculate the data base coordinate system displacement vectors for each face involved. The procedure for interpolating across edges (two face) is as follows: (1) starting at the end of a scan line, project the end point in both coordinate systems; (2) test to see which coordinate system produces a point lying in the face of that coordinate system (i.e., test x and y coordinates against the range of x and y); and (3) interpolate in that system, testing against the range of (x, y) accordingly. When a point exceeds the range of (x, y), change systems and continue. In

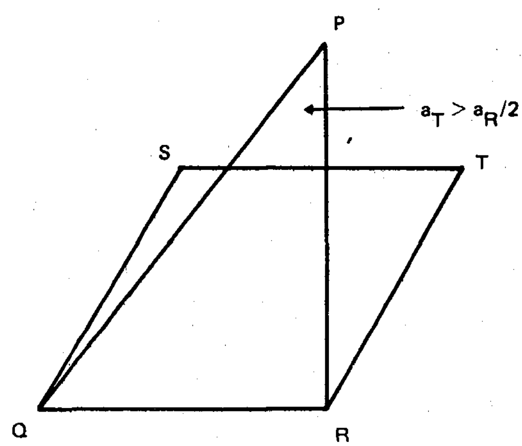

Figure 4-8. Vertex Point P Lies Outside QRST

the case of two faces, it is not necessary to continue testing the range since a scan line will cross only one boundary. The three-face case is similar except that, when the test fails, the other two face systems must be tested to determine which to use, and testing aginst the (x, y) range must continue until two-face boundaries have been crossed.

SECTION 5 - INPUT/OUTPUT MANAGEMENT

5.1 SUMMARY

In considering the design of any large-scale data base system, the efficient design of I/O management facilities is of great importance. This is especially critical in the case of a data base of this nature and magnitude. Up to this point, the data base has been considered in a general sense. This section discusses a specific data base with properties approximating an operational data base.

5.2 EXEMPLARY MODEL

Consider a data base containing whole Earth information for one spectral band consisting of one data item per pixel. As a proposed model, let each item be represented by one byte (6 or 8 bits) and let each scan line contain 1280 pixels at a resolution of 1 nautical mile, constant along the scan line.

The area of the Earth in square nautical miles is approximately

$$4\pi \times \left(3.48 \times 10^{3}\right)^{2} = 1.52 \times 10^{8}$$

The nearest level of division giving coarser resolution is the 12th level

$$6 \times 4^{12} = 6 \times 4096^{2} = 100663296 = 1.01 \times 10^{8}$$

The linear resolution of this data base is

$$\sqrt{\frac{1.52}{1.01}} = 1.22 \text{ nautical miles}$$

Therefore, at this level, 10^8 bytes of storage will be required. A convenient (but not unrealistic) model for disk type storage can be chosen to have six spindles, 256 cylinders/spindle, 16 tracks/cylinder, and 4096 bytes/track. Thus, there will be a total of $6 \times 256 \times 16 \times 4096 = 10^8$ bytes of storage. By splitting the serial data string over such a facility, it is advantageous to choose one face to correspond to one spindle, and a natural record to correspond to one track. Hence, each face would be divided into $256 \times 16 = 4096$ physical records, and each record would represent division of a face at the sixth level $(4096 = 4^6)$. Because there are two bits associated with each level of division, a typical serial string address would then appear as follows:

$$\underbrace{101}_{\text{Face Number}} \quad \underbrace{111011000011}_{\text{Physical Record Address}} \quad \underbrace{101011110011}_{\text{Address Within Record}}$$

Each physical record would thus represent an area 64 units square. This size is of the same order as the typical propagation distance for the interpolation described in Section 4. Figure 5-1 illustrates the relation of various ratios of interpolation block size to I/O record size.

In the case where the two sizes are not identical, and where the blocks and records are not perfectly aligned, a certain amount of redundant I/O transfer is inevitable. The amount of redundant transfer decreases as the ratio of interpolation to record size increases. On the other hand, the number of individual I/O operations will rise if this ratio is obtained by lowering the record size. The optimum value for this ratio will have to be determined by studying the actual machine configuration on which the data base is implemented.

For example, a 64×64 unit record has been chosen to provide reasonable savings in computation. Similarly, it is preferable to adopt the same size

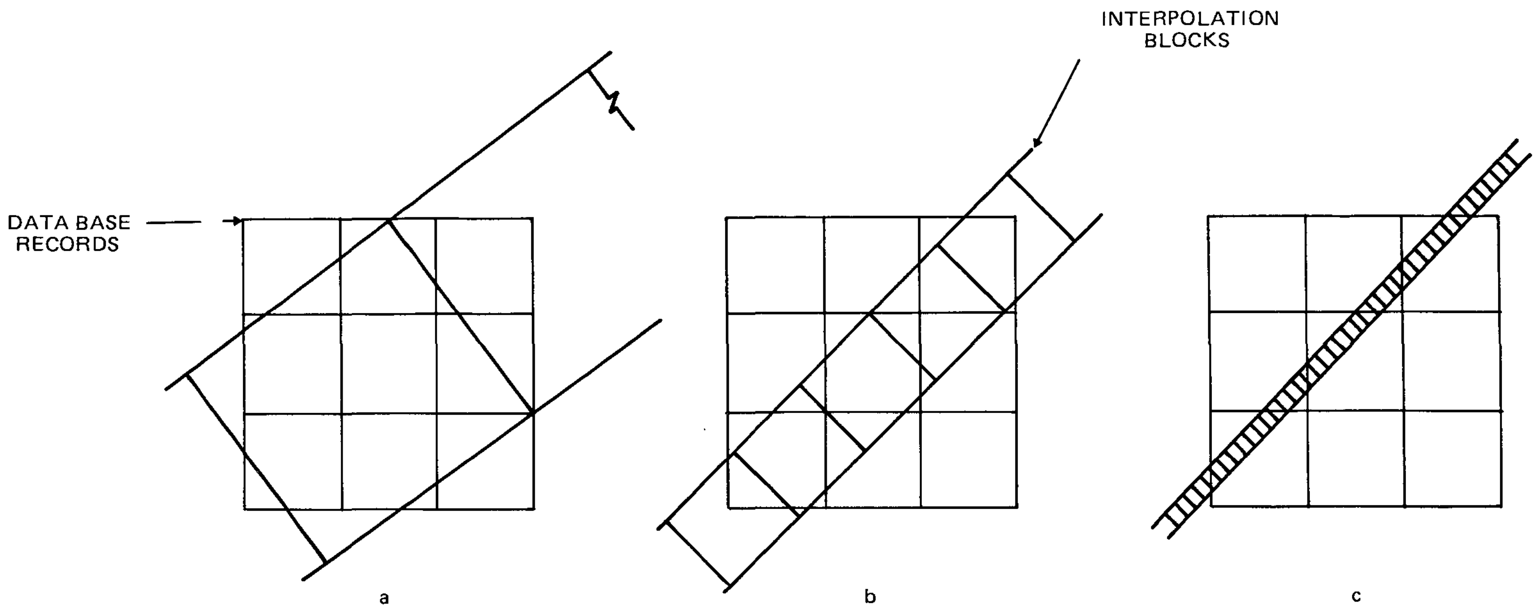

Figure 5-1. Relationship of Interpolation Blocks to I/O Records

interpolation block so that there will be $64 \times 64 = 4096$ interpolations per mapping computation. This ratio is illustrated in Figure 5-1(b). To assure complete coverage of the interpolation block, six records will be required to minimize I/O operations. Consideration must also be given to sufficient core buffer space to store all six records simultaneously, requiring 24,576 bytes of storage. Data will be queued on a first-in first-out (FIFO) basis, meaning that when a record not currently in core is required, it will replace that record which has been in core the longest. Figure 5-2 illustrates the typical result of such a scheme. Records 1 through 6 are resident in core at one time. When record 7 is required, it replaces record 1; record 8 replaces record 2; and so on.

The implementation of such a scheme is accomplished via the use of a push-down stack of fixed depth. The stack consists of a set of pointers to the initial location of a record's core location and the serial number of the record. When a record is required, the stack is searched by record number. If the record is present, the associated pointer locates the record; if the record is not present, its serial number is added to the top of the stack. The record which is pushed out of the bottom of the stack is returned to disk storage and the new record required is read into core in the newly freed locations. The pointer which was associated with the oldest record, now is associated with the new record.

The procedure for accessing the data base thus breaks down into two parts: (1) the calculation of serial addresses, and (2) the retrieval and storage of records implied by the serial addresses. As discussed previously, addresses are calculated in blocks whose size is based on the interpolation range. Taking a scan line of length 1280 pixels and the chosen 64×64 pixel interpolation block, it is required to have $64 \times 1280 = 81,920$ bytes divided into 20 64×64 pixel blocks. Looping over pixels and scan lines in these blocks, calculations will be performed for the serial address as previously discussed. Each serial

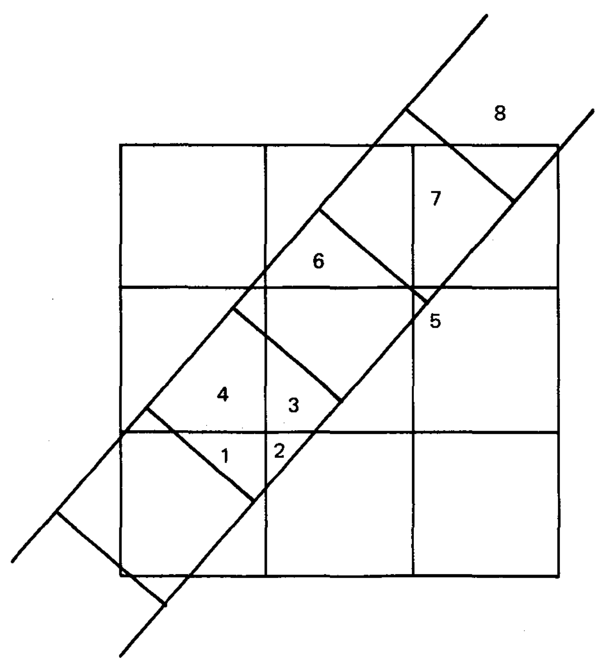

Figure 5-2. Order of First Appearance for I/O Records

address, as it is computed, is compared under a mask with the previous serial address to test for a change in physical record number. If there is no required change in record number, the data item is stored into the core location found by summing the within-record portion of the address and the base address currently in use. On the other hand, if there is a change in the record number, the new record desired is compared with the list of records in core (the stack), and an I/O operation is performed when appropriate. The base address for this new record (as stored in the stack) now becomes the active base address, and the data item is stored in the appropriate location.

The primary features of the preceding scheme are

1. I/O takes place only when a location is not available in the core buffer

2. All memory references may make use of a base address register

3. Testing for correctness of the current record requires only a simple test under mask operation

Examination of Figure 5-2 shows that record changes take place approximately once per scan line or once per 64 address computations. Thus, each interpolation block requires access to two or three records not currently in the stack (say 2.5 as a conservative estimate). The time required for a record change not necessitating an I/O operation is negligible (six comparison operations every 64 address calculations). However, the time required when I/O takes place is considerable.

To calculate the I/O time required, consider the following realistic but conservative set of specifications for a disk storage device:

- Data transfer rate: 500,000 bytes/second
- Average rotational latency: 10 msec
- Average seek time: 30 msec

At the 500K baud transfer rate, one 4096 byte record would take 8 msec. The near-neighbor relationship established by the serial addressing scheme assures that approximately 75 percent of the immediate neighbors of a record will lie on the same cylinder, thus giving an average access time of 15 msec. Using the estimate of 2.5 data base accesses per 64 × 64 block and the fact that each access involves one input and one output operation, it is found to require an I/O time of 5 × (15 + 8) = 115 msec per block or 2.3 seconds per 64 scan line input block. This yields approximately 900 I/O seconds per orbit, comfortably faster than a real-time value of 6000 seconds.

Another way of looking at this I/O requirement is that it would take approximately 2700 seconds or 3/4 hour to fill six faces of 4096 records each. These figures are conservative and could be improved considerably by using faster hardware and by spreading adjacent records over corresponding cylinders of all available spindles. In any case, the I/O requirements are clearly well within the state of the art and would not unduly overload a large-scale, multipurpose computer system.

5.3 HIGHER RESOLUTION DATA BASES AND USE OF SELECTED AREAS

A whole Earth data base at one mile resolution requires 10^8 bytes of storage per spectral band. This number is within the range of current disk storage devices. A higher resolution data base would, however, rapidly exceed the capacities of even the largest disk type storage devices. A data base at 0.1 mile resolution would require 10^{10} bytes, corresponding to literally hundreds of disk packs. Mass storage devices, as described in Appendix C, would allow storage of such an array.

The basic record addressing scheme discussed in Section 5.2 requires online access to a full six-faced data base. For purposes of storing high resolution data, some scheme for storing selected areas is necessary. The simplest

method of accomplishing this is by using an indirect addressing scheme. Figure 5-3 shows a face divided at the fourth level.

Blocks of records corresponding to these fourth-level areas can be stored for regions of interest. An array containing a logical flag specifying the existence of records would be stored along with a pointer indicating the real starting record number for the block. For a fourth-level division, two $16^2 = 256$ arrays will be required to store the flags and pointers.

The I/O management module in the program would first do a shift to determine the storage block number, check this against the flag array, and accordingly access the record by adding the within-block portion of the record number to the starting record number for that block.

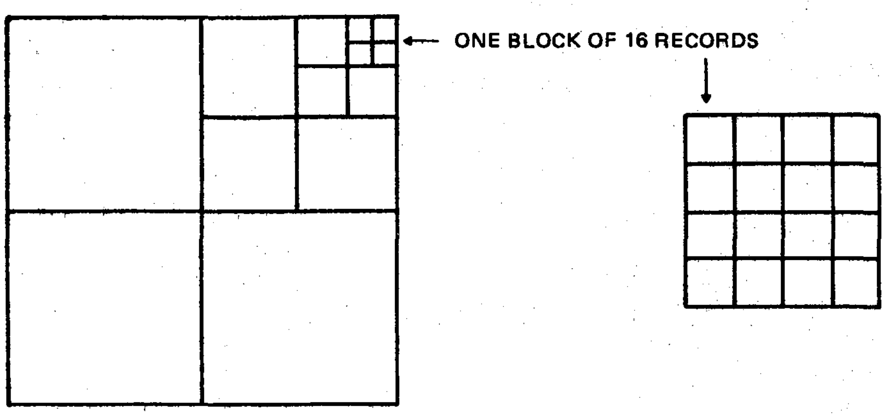

Figure 5-3. Face Divided at Fourth Level for Windowing

SECTION 6 - DATA RETRIEVAL

6.1 SUMMARY

This section is concerned with accessing data from the data base for the purposes of (1) numerical processing and (2) visual display. These are considered to be the two main needs of the contemplated user who would be interested in using the system.

6.2 DATA ACCESS FOR USER NEEDS

For the two purposes mentioned above, retrieval in physical record units would usually make sense. For most visual purposes, shape distortion would be serious only in the vertex regions where 120-degree angles map into 90-degree angles. This distortion could be minimized by using an overlay grid.

For numeric purposes, the equal-area property of the data base should make the face coordinate systems extraordinarily useful. Most applications will, however, require a conversion to a more normal two-dimensional mapping (raster) scheme. An algorithm which accomplishes this conversion is presented in the source listing attached as Appendix E.9.

For visual purposes, a useful display scheme would be to display an unfolded cube as in Figure 1-6, featuring continental outlines and a data base coordinate grid. The user could access any record or group of records by indicating the corners of the desired area with a light pen or some other device. The system would then retrieve the indicated records, convert to raster form, and display. Precalculated geographic or other grids could be overlayed on the data to provide referencing.

Alternatively, the data base system could be reconverted into a geographic grid system by reversing the mapping and interpolation procedure and applying the transformation:

$$\xi = f(x, y)$$

$$\eta = f(y, x)$$

Interpolation and I/O considerations would be similar to those described previously except that no output operations would be necessary.

A particularly rapid scheme for display would involve the use of a special piece of hardware, similar to the x, y register summing device discussed earlier. A serial address decoder accessing storage via a direct memory access (DMA) type device would allow display of a selected set of records with no central processor activity other than I/O. In a distributed system, such as that described in Appendix D, this device would be part of a stand-alone display device accessing data on a staging device.

SECTION 7 - RESULTS OF STUDY

7.1 SUMMARY

The primary result of this study is the conclusion that implementation of a Quadrilateralized Spherical Cube Data Base is feasible using current large-scale computer systems. Numerical results supporting this conclusion are presented below.

7.2 NUMERIC RESULTS OF MAPPING FUNCTION

The transformation function $f(x , y)$ described in Section 2.2 was studied intensively. A FORTRAN program was written to minimize the residual function ϕ. The function was evaluated at 36 locations on a regular grid covering one quadrant of a face. (The symmetry of Equation (2-14) permits the generalization to the other three quadrants of the face.)

Table 7-1 illustrates the very nearly equal-area nature of the projection. The table presents the ratio

$$\left[\frac{\left(\frac{\partial\xi}{\partial x}\frac{\partial\eta}{\partial y} - \frac{\partial\xi}{\partial y}\frac{\partial\eta}{\partial x}\right)}{\lambda^2 r_o^2\left(1 + \frac{\xi^2}{r_o^2} + \frac{\eta^2}{r_o^2}\right)^{3/2}}\right]$$

which appears in Equation (2-26). In other words, it presents the transformed area at a given point normalized to the projected area at the center of a face. For an exact transformation, this ratio is identically equal to unity. The calculated root mean-square deviation over the entire face is found to be 0.000103.

The function $f^*(\xi , \eta)$ was obtained from $(f(x , y)$ as indicated in Section 2.3. The procedure employed was to minimize the residual function ϕ^* calculated on the same regular grid covering one equadrant of a face.

Table 7-1. Tabulation of Direct Mapping Function Accuracy

	CENTER 0.0	0.2	0.4	0.6	0.8	1.0
0.0	1.0	1.0001	0.9999	0.9999	1.0001	1.0
0.2	1.0001	1.0002	1.0001	1.0001	1.0001	1.0002
0.4	0.0	1.0001	1.0	1.0001	0.9999	1.0002
0.6	0.9999	1.0001	1.0001	1.0	0.9999	1.0001
0.8	1.0001	1.0001	0.9999	0.9999	1.0001	0.9998
1.0	1.0	1.0002	1.0002	1.0001	0.9998	1.0 VERTEX

Table 7-2 presents the results of this minimization, showing the values of

$$\sqrt{\left\{\left[x - f^*(f(x, y), f(y, x))\right]^2 + \left[y - f^*(f(y, x), f(x, y))\right]^2\right\}}$$

over the grid. For an exact inverse transformation, these values are identically equal to zero. The RMS deviation of these values is found to be 0.012.

7.3 NUMERIC RESULTS FROM TRANSFORMATION AND FAST-FILLING ALGORITHM TESTS

A FORTRAN-coded version of the $(\xi, \eta) \rightarrow (x, y)$ transformation algorithm was tested. This algorithm requires the following arithmetic operations to map one point:

Arithmetic comparisons	3
Integer multiplication	1
Integer subtraction	1
Real multiplications	51
Real divisions	5
Real additions	26
Real subtractions	2

In this form, the algorithm requires approximately 150 μsec to execute on the IBM S/360-91 used for these tests.

The interpolation and tests used in the simple fast-filling method require 30 μsec per point to calculate the serial address and store one point into a data base record. This time could be reduced somewhat by coding in assembler language, but is reasonably close to the irreducible minimum.

The two vital figures in this study are the 30 μsec calculation time per data point and the average I/O time per point. Using the value of 115 msec per 64×64 block obtained in Section 5, this is also approximately 30 μsec per

Table 7-2. Tabulation of Inverse Mapping Function Accuracy

	CENTER 0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.0	0.003	0.007	0.020	0.008	0.0
0.2	0.003	0.008	0.009	0.019	0.016	0.027
0.4	0.007	0.009	0.016	0.011	0.015	0.005
0.6	0.020	0.019	0.011	0.013	0.015	0.025
0.8	0.008	0.016	0.015	0.015	0.008	0.018
1.0	0.0	0.027	0.005	0.025	0.018	0.0 VERTEX

point. Overhead from other operations should not contribute significantly since all operations will be done in blocks of the order 10^3. For a 1280 point scan line, computations and I/O times of the order of 0.02 second per scan line are expected.

For a 100-minute period satellite, the ground track speed is about 3.5 nautical miles/second. At 1 mile resolution, this means computation would be taking place at about 15 times the data rate. At 0.1 mile resolution, computation would be 1.5 times faster than real time.

The above figures are approximate and refer to execution times on the IBM S/360-91; they are conservative and could be improved upon by utilizing special hardware or by maximizing coding efficiency. They indicate that the implementation of a Quadrilateralized Spherical Cube Data Base is feasible on current generation, large-scale computers, particularly at coarser resolutions.

APPENDIX A - DETAILS OF MAPPING FUNCTION

A.1 DETAILS OF DIRECT MAPPING FUNCTION

Without any loss of generality, Equation (2-22) may be re-written as

$$\xi = f(x,\ y) = \gamma x + \frac{(1-\gamma)}{r_o^2}\, x^3 + x\left(r_o^2 - x^2\right)\left[y^2 g(x,\ y) + x^2 h(x)\right] \qquad \text{(A-1)}$$

where $g(x\ ,\ y)$ and $h(x)$ are even functions of their arguments. Differentiation with respect to x yields

$$\begin{aligned}\frac{\partial \xi}{\partial x} = \gamma &+ \frac{3(1-\gamma)}{r_o^2}\, x^2 + y^2\left(r_o^2 - x^2\right) g(x,\ y) - 2x^2y^2 g(x,\ y) \\ &+ xy^2\left(r_o^2 - x^2\right)\frac{\partial g}{\partial x} + 3x^2\left(r_o^2 - x^2\right) h(x) - 2x^4 h(x) \qquad \text{(A-2)} \\ &+ x^3\left(r_o^2 - x^2\right)\frac{dh}{dx}\end{aligned}$$

Substitution of the condition in Equation (2-19) into the above equation yields

$$\mu = \left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o \\ y=r_o}} = 3 - 2\gamma - 2r_o^4 g(r_o,\ r_o) - 2r_o^4 h(r_o) \qquad \text{(A-3)}$$

For convenience, let δ and ω be defined by

$$\begin{aligned}\delta &= g(r_o,\ r_o) \\ \omega &= h(r_o)\end{aligned} \qquad \text{(A-4)}$$

It follows that $g(x, y)$ and $h(x)$ can be expressed as

$$g(x, y) = \delta + \left(r_o^2 - y^2\right) p(x, y) + \left(r_o^2 - x^2\right) s(x, y) \tag{A-5}$$

$$h(x) = \omega + \left(r_o^2 - x^2\right) q(x) \tag{A-6}$$

where $p(x, y)$, $q(x)$, and $s(x, y)$ are even functions of their arguments. Moreover, from Equations (A-3) and (A-4), ω can be expressed in terms of δ by the equation

$$\omega = \frac{1}{2 r_o^4} \left(3 - 2\gamma - \mu - 2 r_o^4 \delta\right) \tag{A-7}$$

Next, at the point $(\xi = x = r_o , \eta = y = 0)$, it is noted that Equation (2-12) yields

$$\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o\\y=0}} \cdot \left.\frac{\partial \eta}{\partial y}\right|_{\substack{x=r_o\\y=0}} - \left.\frac{\partial \xi}{\partial y}\right|_{\substack{x=r_o\\y=0}} \cdot \left.\frac{\partial \eta}{\partial x}\right|_{\substack{x=r_o\\y=0}} = 2\sqrt{2}\gamma^2 \tag{A-8}$$

However, from the symmetry and boundary conditions imposed on the mapping function $f(x, y)$, it is evident from Figure 1-4 that

$$\left.\frac{\partial \xi}{\partial y}\right|_{\substack{x=r_o\\y=0}} = 0$$

$$\left.\frac{\partial \eta}{\partial x}\right|_{\substack{x=r_o\\y=0}} = 0 \tag{A-9}$$

Moreover, from the symmetry of Equation (2-14)

$$\left.\frac{\partial \eta}{\partial y}\right|_{\substack{x=r_o \\ y=0}} = \left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0 \\ y=r_o}} \tag{A-10}$$

Consequently, Equation (A-8) becomes

$$\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o \\ y=0}} \cdot \left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0 \\ y=r_o}} = 2\sqrt{2}\gamma^2 \tag{A-11}$$

Equations (A-2), (A-5), (A-6), and (A-7) yield

$$\begin{aligned} \left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o \\ y=0}} &= 3 - 2\gamma - 2r_o^4 h(r_o) \\ &= 3 - 2\gamma - 2r_o^4 \omega \\ &= \mu + 2r_o^4 \delta \end{aligned} \tag{A-12}$$

$$\begin{aligned} \left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0 \\ y=r_o}} &= \gamma + r_o^4 g(0, r_o) \\ &= \gamma + r_o^4 \left[\delta + r_o^2 s(0, r_o)\right] \end{aligned} \tag{A-13}$$

Next, it is noted that the number of terms in $g(x, y)$ given by Equation (A-5) will be minimized by choosing

$$s(x, y) \equiv 0 \tag{A-14}$$

This will also eliminate the terms $s(0, r)$ in Equation (A-13), which then becomes

$$\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0 \\ y=r_o}} = \gamma + r_o^4 \delta \tag{A-15}$$

Substitution of Equations (A-12) and (A-15) into Equation (A-11) yields

$$\left(\mu + 2 r_o^4 \delta\right)\left(\gamma + r_o^4 \delta\right) = 2\sqrt{2}\gamma^2 \tag{A-16}$$

or, equivalently,

$$2 r_o^8 \delta^2 + (\mu + 2\gamma) r_o^4 \delta + (\mu\gamma - 2\sqrt{2}\gamma^2) = 0 \tag{A-17}$$

Solution of this equation for $r_o^4 \delta$ yields

$$r_o^4 \delta = \frac{-B \pm \sqrt{(B^2 - 4AC)}}{2A} \tag{A-18}$$

where

$$A = 2$$

$$B = (\mu + 2\gamma) > 0 \tag{A-19}$$

$$C = (\mu\gamma - 2\sqrt{2}\gamma^2) < 0$$

Consideration of Figure 1-4 reveals that $\delta > 0$. Hence, only the plus (+) sign is retained in Equation (A-17). Thus, the following expression is obtained for δ:

$$\delta = \frac{1}{4 r_o^4} \left[- (\mu + 2\gamma) + \sqrt{(\mu^2 - 4\mu\gamma + 4\gamma^2 + 16\sqrt{2}\gamma^2)} \right] \tag{A-20}$$

$$= 0.79048\ 64491\ 208$$

The value of ω is obtained by using Equation (A-7):

$$\omega = \frac{1}{2 r_o^4} \left(3 - 2\gamma - \mu - 2 r_o^4 \delta \right) \tag{A-21}$$

$$= -1.2254\ 41487\ 984$$

Finally, in view of the above discussion, the direct mapping function $f(x, y)$ can be chosen to have the form

$$f(x, y) = \gamma x + \frac{(1 - \gamma)}{r_o^2} x^3 + xy^2 \left(r_o^2 - x^2\right) \left[\delta + \left(r_o^2 - y^2\right) \sum_{\substack{i \geq 0 \\ j \geq 0}} c_{ij} x^{2i} y^{2j} \right]$$

$$+ x^3 \left(r_o^2 - x^2\right) \left[\omega + \left(r_o^2 - x^2\right) \sum_{i \geq 0} d_i x^{2i} \right] \tag{A-22}$$

A.2 DETAILS OF INVERSE MAPPING FUNCTION

From Equation (2-12) and the following property of Jacobians

$$J\left(\frac{\xi,\ \eta}{x,\ y}\right) \cdot J\left(\frac{x,\ y}{\xi,\ \eta}\right) \equiv 1 \tag{A-23}$$

it follows that the inverse transformation

$$\begin{aligned} x &= x(\xi,\ \eta) \\ y &= y(\xi,\ \eta) \end{aligned} \tag{A-24}$$

satisfies the following equation

$$\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} = \frac{1}{\gamma^2\left(1 + \frac{\xi^2}{r_o^2} + \frac{\eta^2}{r_o^2}\right)^{3/2}} \tag{A-25}$$

Because the direct transformation in Equation (2-14) is symmetrical, the inverse transformation is also symmetrical, i.e.,

$$\begin{aligned} x &= f^*(\xi,\ \eta) \\ y &= f^*(\eta,\ \xi) \end{aligned} \tag{A-26}$$

Proceeding as in Section 2.2, it is seen that the following four conditions are the counterparts of Equations (2-16) through (2-19)

$$f^*(0,\ \eta) = 0 \tag{A-27}$$

$$f^*(r_o,\ \eta) = r_o \qquad \text{(A-28)}$$

$$\left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=0\\ \eta=0}} = \left.\frac{\partial y}{\partial \eta}\right|_{\substack{\xi=0\\ \eta=0}} = \frac{1}{\gamma} \equiv \gamma^* \qquad \text{(A-29)}$$

$$\left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=r_o\\ \eta=r_o}} = \left.\frac{\partial \eta}{\partial y}\right|_{\substack{\xi=r_o\\ \eta=r_o}} = \frac{1}{\mu} \equiv \mu^* \qquad \text{(A-30)}$$

Thus, the first three conditions above may be incorporated into the following expansion which is the analog of Equation (A-1):

$$x = f^*(\xi,\ \eta) = \gamma^* \xi + \frac{(1-\gamma^*)}{r_o^2}\,\xi^3 + \xi\left(r_o^2 - \xi^2\right)\left[\eta^2 g^*(\xi,\ \eta) + \xi^2 h^*(\xi)\right] \qquad \text{(A-31)}$$

where

$$g^*(\xi,\ \eta) = \delta^* + \left(r_o^2 - \eta^2\right) p^*(\xi,\ \eta) + \left(r_o^2 - \xi^2\right) s^*(\xi,\ \eta) \qquad \text{(A-32)}$$

$$h^*(\xi) = \omega^* + \left(r_o^2 - \xi^2\right) q^*(\xi) \qquad \text{(A-33)}$$

$$\begin{aligned} \delta^* &\equiv g^*(r_o,\ r_o) \\ \omega^* &\equiv h^*(r_o) \end{aligned} \qquad \text{(A-34)}$$

$p^*(\xi,\ \eta)$, $q^*(\xi)$, and $s^*(\xi,\ \eta)$ are even functions of their arguments.

The fourth condition, i.e., Equation (A-30), yields

$$\omega^* = \frac{1}{2\, r_o^4} \left(3 - 2\gamma^* - \mu^* - 2\, r_o^4\, \delta^*\right) \tag{A-35}$$

which is the analog of Equation (A-7).

Next, by differentiating Equation (A-31) and using Equations (A-32) through (A-35), the following analogs of Equations (A-12) and (A-15) are obtained:

$$\begin{aligned} \left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=r_o \\ \eta=0}} &= 3 - 2\gamma^* - 2\, r_o^4\, \omega^* \\ &= \mu^* + 2\, r_o^4\, \delta^* \end{aligned} \tag{A-36}$$

$$\left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=0 \\ \eta=r_o}} = \gamma^* + r_o^4 \left[\delta^* + r_o^2\, s^*(0,\ r_o)\right] \tag{A-37}$$

However, consideration of Figure 1-4 reveals that the following expressions are also valid:

$$\left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=r_o \\ \eta=0}} = \left[\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=r_o \\ y=0}}\right]^{-1} = \frac{1}{\mu + 2\, r_o^4\, \delta} \equiv \mu_1^* \tag{A-38}$$

$$\left.\frac{\partial x}{\partial \xi}\right|_{\substack{\xi=0 \\ \eta=r_o}} = \left[\left.\frac{\partial \xi}{\partial x}\right|_{\substack{x=0 \\ y=r_o}}\right]^{-1} = \frac{1}{\gamma + r_o^4\, \delta} \equiv \gamma_1^* \tag{A-39}$$

By equating the corresponding expressions above and then solving for δ^* and $s^*(0\ ,\ r_o)$, it is found that

$$\delta^* = \frac{(\mu_1^* - \mu^*)}{2\, r_o^4} \tag{A-40}$$

$$s^*(0,\ r_o) = \frac{1}{r_o^2}\left[\frac{(\gamma_1^* - \gamma^*)}{r_o^4} - \delta^*\right] \tag{A-41}$$

For convenience, let δ_1^* be defined by

$$\delta_1^* = s^*(0,\ r_o) \tag{A-42}$$

Then, it follows that $s^*(\xi,\ \eta)$ can be expressed as

$$s^*(\xi,\ \eta) = \delta_1^* + \xi^2\, u^*(\xi) + \left(r_o^2 - \eta^2\right) v^*(\xi,\ \eta) \tag{A-43}$$

where $u^*(\xi)$ and $v^*(\xi\ ,\ \eta)$ are even functions of their arguments. For the purpose of expanding $x = f^*(\xi\ ,\ \eta)$ and also satisfying the six conditions described by Equations (A-27) through (A-30) and (A-38) through (A-39), it may be verified that it suffices to let

$$\begin{aligned} u^*(\xi) &\equiv 0 \\ v^*(\xi,\ \eta) &\equiv 0 \end{aligned} \tag{A-44}$$

Finally, in view of the above discussion, the inverse mapping function $f^*(\xi, \eta)$ can be chosen to have the form

$$f^*(\xi, \eta) = \gamma^* \xi + \frac{(1 - \gamma^*)}{r_o^2} \xi^3 + \xi \eta^2 \left(r_o^2 - \xi^2\right)\left[\delta^* + \delta_1^*\left(r_o^2 - \xi^2\right)\right.$$

$$\left. + \left(r_o^2 - \eta^2\right) \sum_{\substack{i \geq 0 \\ j \geq 0}} c_{ij}^* \xi^{2i} \eta^{2j}\right] + \xi^3\left(r_o^2 - \xi^2\right)\left[\omega^*\right. \qquad \text{(A-45)}$$

$$\left. - \left(r_o^2 - \xi^2\right) \sum_{i \geq 0} d_i^* \xi^{2i}\right]$$

APPENDIX B - DEPENDENCE OF SCAN ELEMENT LOCATION ON ORBIT AND ATTITUDE OF SATELLITE

B.1 THE CASE OF ZERO PITCH, ROLL, AND YAW

Consider an Earth-centered "auxiliary" orbital coordinate system (x^A, y^A, z^A), as illustrated in Figure B-1, such that the satellite is always located on the y^A-axis and the scan angle σ is always in the (y^A, z^A)-plane. Then, the z^A-axis is always perpendicular to the plane of the orbit, and the x^A- and y^A-axes move within the orbital plane, rotating about the z^A-axis.

It is noted that the point of intersection, Q, is such that $|\mu| < \pi/2$ so that $\cos\mu$ is always given by

$$\cos\mu = \sqrt{(1 - \sin^2\mu)} \tag{B-1}$$

From the sine formula for ΔOPQ, it follows that

$$\frac{\sin\sigma}{R} = \frac{\sin(\pi - \sigma - \mu)}{(R + h)} \tag{B-2}$$

Using the relation

$$\sin(\pi - \sigma - \mu) = \sin(\sigma + \mu) = \sin\sigma\cos\mu + \cos\sigma\sin\mu \tag{B-3}$$

Equation (B-2) becomes

$$\frac{\sin\sigma}{R} = \frac{\sin\sigma\cos\mu + \cos\sigma\sin\mu}{(R + h)} \tag{B-4}$$

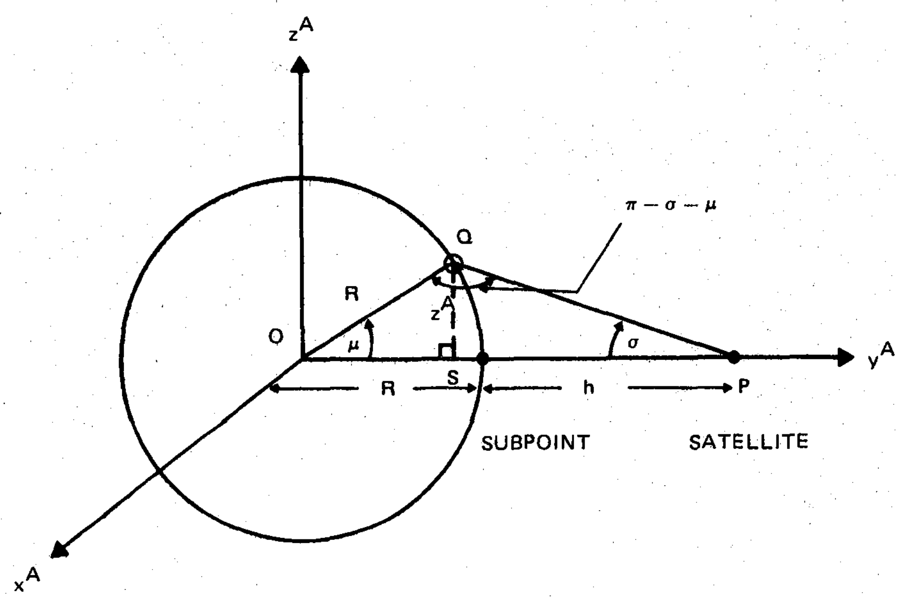

Figure B-1. Auxiliary Orbital Coordinate System

or, equivalently,

$$\cos\mu + \cot\sigma \sin\mu = \frac{(R + h)}{R} \qquad \text{(B-5)}$$

Using Equation (B-1), this becomes

$$\sqrt{(1 - \sin^2\mu)} + \cot\sigma \sin\mu = \frac{(R + h)}{R} \qquad \text{(B-6)}$$

However, it is noted from ΔOQS that

$$\sin\mu = \frac{z_A}{R} \qquad \text{(B-7)}$$

Hence, Equation (B-6) becomes

$$\sqrt{\left(1 - \frac{z_A^2}{R^2}\right)} = \frac{(R + h)}{R} - \cot\sigma \frac{z_A}{R} \qquad \text{(B-8)}$$

or, on squaring,

$$1 - \frac{z_A^2}{R^2} = \frac{(R + h)^2}{R^2} - 2\,\frac{(R + h)}{R} \cot\sigma \frac{z_A}{R} + \cot^2\sigma \frac{z_A^2}{R^2} \qquad \text{(B-9)}$$

or

$$(1 + \cot^2\sigma)\frac{z_A^2}{R^2} - \left[2\,\frac{(R + h)}{R}\cot\sigma\right]\frac{z_A}{R} + \left[\frac{(R + h)^2}{R^2} - 1\right] = 0 \qquad \text{(B-10)}$$

Solution of this equation yields

$$\frac{z^A}{R} = \frac{-B \pm \sqrt{(B^2 - 4AC)}}{2A} \tag{B-11}$$

where

$$\begin{aligned} A &= 1 + \cot^2 \sigma = \csc^2 \sigma \\ B &= -2 \frac{(R + h)}{R} \cot \sigma \\ C &= \frac{(R + h)^2}{R^2} - 1 \end{aligned} \tag{B-12}$$

Since the point of intersection, Q , is on the closer side to P , its z^A-coordinate is given by Equation (B-11) in which only the minus (-) sign is retained. Thus, Equation (B-11) becomes

$$\frac{z^A}{R} = \frac{(R + h)}{R} \sin \sigma \cos \sigma - \sin^2 \sigma \sqrt{\left[\csc^2 \sigma - \frac{(R + h)^2}{R^2}\right]} \tag{B-13}$$

Also, from Figure B-1 it is noted that

$$y^A = R \cos \mu = \sqrt{(R^2 - {z^A}^2)} \tag{B-14}$$

and

$$x^A = 0 \tag{B-15}$$

Next, consider the Earth-centered inertial coordinate system (x^I, y^I, z^I), as shown in Figure B-2.

Then, it is well-known that the Euler angles ϕ, θ, and ψ are given by

$$\phi = \Omega \tag{B-16}$$

$$\theta = i \tag{B-17}$$

$$\psi = \omega + v - \pi/2 \tag{B-18}$$

where i = inclination of satellite orbit

Ω = longitude of ascending node

ω = argument of perigee

v = true anomaly

The coordinates of point Q are denoted by (x^I, y^I, z^I) in the two coordinate systems. For convenience, let

$$\begin{aligned} \vec{r}^I &\equiv (x^I, y^I, z^I) \\ \vec{r}^A &\equiv (x^A, y^A, z^A) \end{aligned} \tag{B-19}$$

Then, the vectors $\vec{r}^I$ and $\vec{r}^A$ are related by

$$\vec{r}^I = A^{-1} \vec{r}^A \tag{B-20}$$

where A^{-1} is the matrix given by Equation (4-47) of Reference 2:

$$A^{-1} = \tilde{A} = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & \sin\theta\sin\phi \\ \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & -\sin\theta\cos\phi \\ \sin\theta\sin\psi & \sin\theta\cos\psi & \cos\theta \end{pmatrix} \tag{B-21}$$

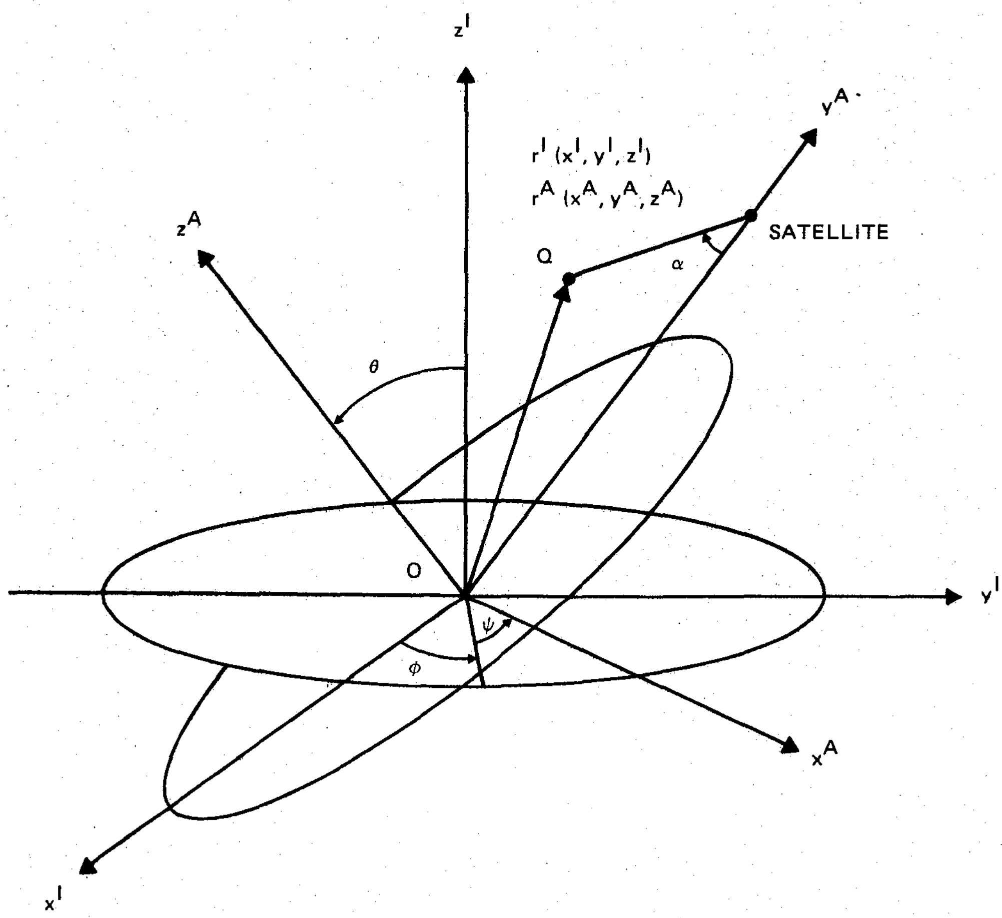

Figure B-2. Inertial Coordinate System

Hence, from Equations (B-15), (B-20), and (B-21), it follows that

$$x^I = (-\sin\psi \cos\phi - \cos\theta \sin\phi \cos\psi)\, y^A + (\sin\theta \sin\phi)\, z^A \tag{B-22}$$

$$y^I = (-\sin\psi \sin\phi + \cos\theta \cos\phi \cos\psi)\, y^A + (-\sin\theta \cos\phi)\, z^A \tag{B-23}$$

$$z^I = (\sin\theta \cos\psi)\, y^A + (\cos\theta)\, z^A \tag{B-24}$$

where y^A, z^A, ϕ, θ, and ψ are given by Equations (B-14), (B-13), (B-16), (B-17), and (B-18), respectively. Hence, the coordinates (x^I, y^I, z^I) of the point Q are obtained.

Finally, consider the uniformly, Earth-centered rotating coordinate system (x^R, y^R, z^R) as shown in Figure B-3.

It is easily shown that

$$x^R = x^I \cos\bar{\omega}t + y^I \sin\bar{\omega}t \tag{B-25}$$

$$y^R = -x^I \sin\bar{\omega}t + y^I \cos\bar{\omega}t \tag{B-26}$$

$$z^R = z^I \tag{B-27}$$

in which $t = 0$ when the two coordinate systems coincide. Thus, the coordinates of a scan element on a rotating Earth are obtained in terms of the satellite orbital position, satellite attitude, and scan angle.

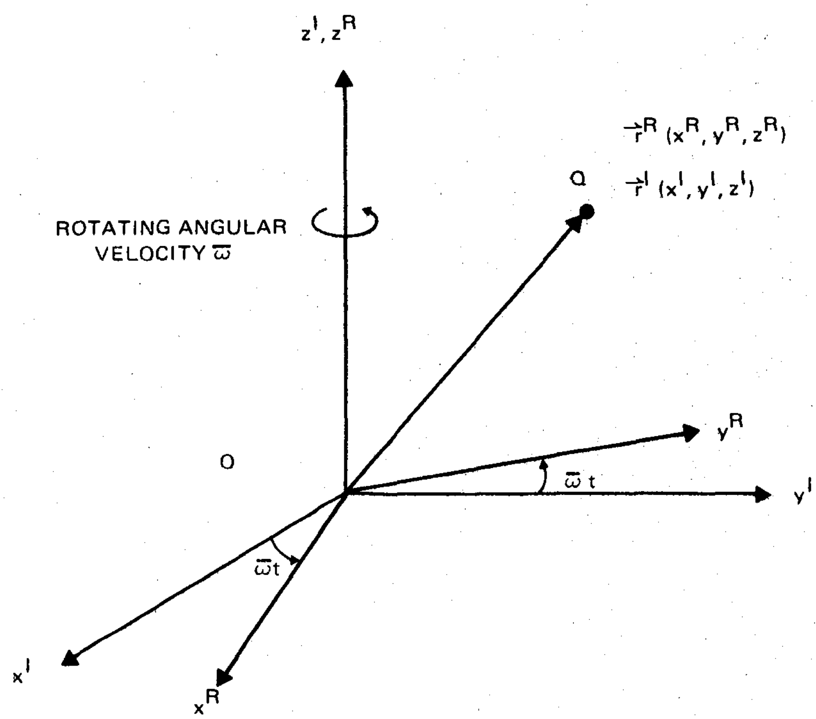

Figure B-3. Uniformly Rotating Coordinate System

B.2 THE CASE OF GENERAL PITCH, ROLL, AND YAW

Let (x^A, y^A, z^A), (x^I, y^I, z^I), and (x^R, y^R, z^R), respectively denote the "auxiliary" frame, the Earth-centered inertial frame, and the Earth-centered rotating frame of reference as defined previously.

Consider an Earth-centered orbital coordinate system (x^o, y^o, z^o) defined by the unit vectors

$$\hat{x}^o = -\hat{x}^A$$
$$\hat{y}^o = -\hat{z}^A \qquad \text{(B-28)}$$
$$\hat{z}^o = -\hat{y}^A$$

Figure B-4 illustrates the orientation of the orbital coordinate system with respect to the "auxiliary" orbital coordinate system.

Let

$$\vec{r}^A \equiv (x^A, y^A, z^A)$$
$$\vec{r}^o \equiv (x^o, y^o, z^o) \qquad \text{(B-29)}$$

Hence, from Figure B-4, it is seen that

$$\begin{bmatrix} x^A \\ y^A \\ z^A \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^o \\ y^o \\ z^0 \end{bmatrix} \qquad \text{(B-30)}$$

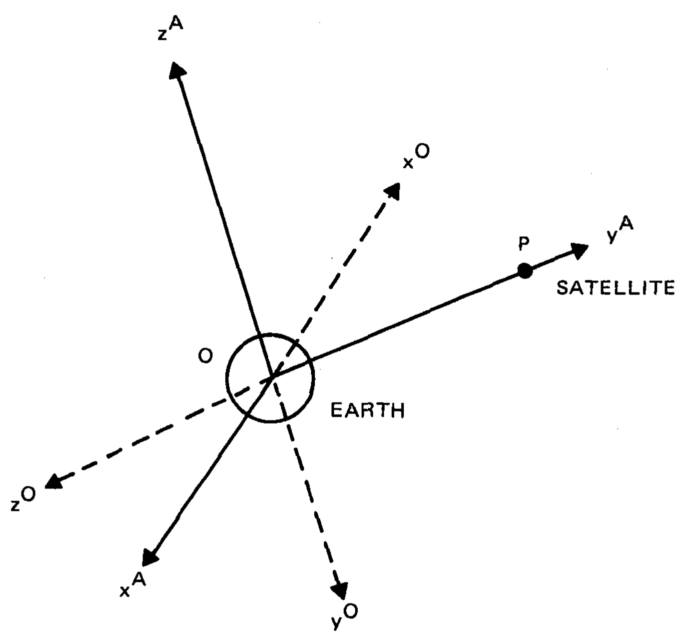

Figure B-4. Orbital Coordinate System

Therefore, the transformation

$$\vec{r}^{A} = B\vec{r}^{o} \tag{B-31}$$

has the matrix B given by

$$B = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{B-32}$$

It is noted that the satellite pitch, roll, and yaw axes are obtained by a parallel translation of the (x^{o}, y^{o}, z^{o}) coordinate system from O to P.

Next, consider a body-fixed coordinate system (x^{B}, y^{B}, z^{B}) on the satellite. Let α, β, and γ respectively denote the roll, pitch, and yaw angles which transform a vector from the orbital to the body-fixed frame. That is, if

$$\begin{aligned} \vec{r}^{o} &\equiv (x^{o}, y^{o}, z^{o}) \\ \vec{r}^{B} &\equiv (x^{B}, y^{B}, z^{B}) \end{aligned} \tag{B-33}$$

then

$$\vec{r}^{B} = C\vec{r}^{o} \tag{B-34}$$

where

$$C = \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma & -\sin\beta\cos\gamma \\ +\sin\beta\sin\alpha\sin\gamma & & +\cos\beta\sin\alpha\sin\gamma \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma & \sin\beta\sin\gamma \\ +\sin\beta\sin\alpha\cos\gamma & & +\cos\beta\sin\alpha\cos\gamma \\ \sin\beta\cos\alpha & -\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \quad \text{(B-35)}$$

Since C is an orthogonal matrix,

$$C^{-1} = C^{T} \quad \text{(B-36)}$$

Hence, from Equation (B-35), the following matrix is obtained for C^{-1} :

$$C^{-1} = \begin{bmatrix} C_\beta C_\gamma & -C_\beta S_\gamma & S_\beta C_\alpha \\ +S_\beta S_\alpha S_\gamma & +S_\beta S_\alpha C_\gamma & \\ C_\alpha S_\gamma & C_\alpha C_\gamma & -S_\alpha \\ -S_\beta C_\gamma & S_\beta S_\gamma & C_\beta C_\alpha \\ +C_\beta S_\alpha S_\gamma & +C_\beta S_\alpha C_\gamma & \end{bmatrix} \quad \text{(B-37)}$$

where the abbreviation S_α and C_α are used to denote $\sin\alpha$ and $\cos\alpha$, respectively.

Let $\hat{n}$ denote the unit vector along the scan ray which is assumed to be in the (y^B, z^B) -plane. The scan angle, σ, is defined as in Figure B-5.

Then, in the body-fixed system, the scan ray unit vector is given by

$$\hat{n} = \hat{n}^B = -\sin\sigma\hat{y}^B + \cos\sigma\hat{z}^B \quad \text{(B-38)}$$

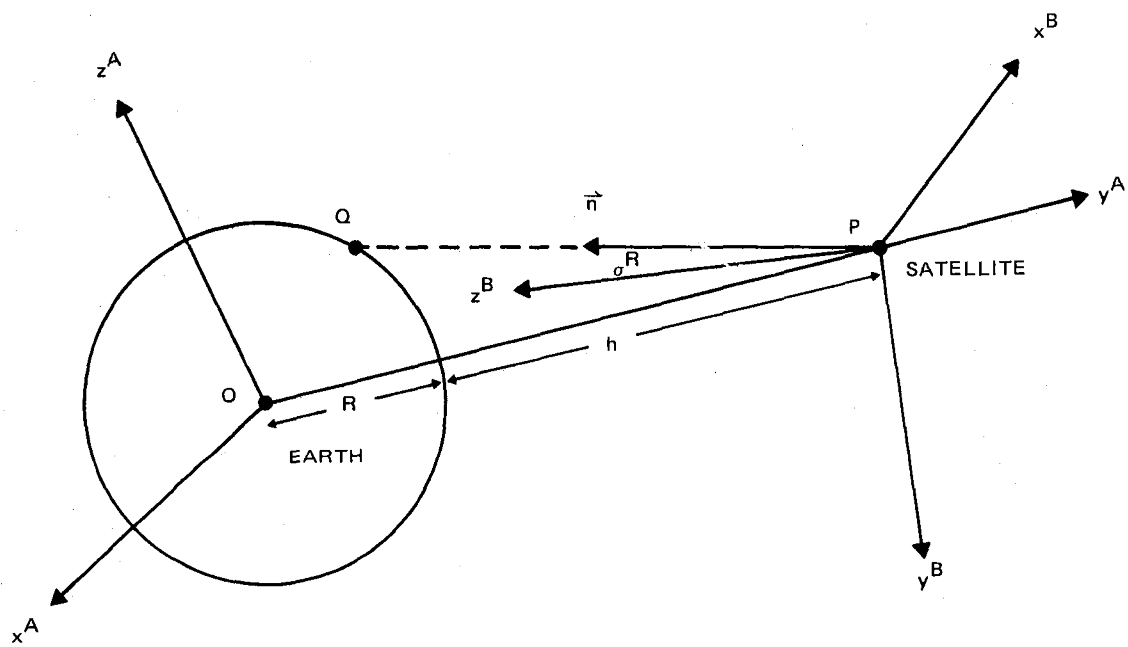

Figure B-5. Satellite Scanning Earth

In general, if the scan ray is not in the (y^B, z^B)-plane, then $\hat{n}^B$ is given by

$$\hat{n} = \hat{n}^B = \cos \sigma_x \hat{x}^B + \cos \sigma_y \hat{y}^B + \cos \sigma_z \hat{z}^B \tag{B-39}$$

where σ_x, σ_y, and σ_z are the angles between $\hat{n}$ and the respective body-fixed coordinate axes.

It follows from Equations (B-31) and (B-34), that in the orbital and the auxiliary orbital systems, the scan ray unit vector is given by

$$\hat{n}^o = C^{-1} \hat{n}^B \tag{B-40}$$

$$\hat{n}^A = B \hat{n}^o \tag{B-41}$$

so that

$$\hat{n}^A = B C^{-1} \hat{n}^B \tag{B-42}$$

From Equations (B-32), (B-37), and (B-38), it follows that

$$B C^{-1} = \begin{bmatrix} -C_\beta C_\gamma - S_\beta S_\alpha S_\gamma & C_\beta S_\gamma - S_\beta S_\alpha C_\gamma & -S_\beta C_\alpha \\ S_\beta C_\gamma - C_\beta S_\alpha S_\gamma & -S_\beta S_\gamma - C_\beta S_\alpha C_\gamma & -C_\beta C_\alpha \\ -C_\alpha S_\gamma & -C_\alpha C_\gamma & S_\alpha \end{bmatrix} \tag{B-43}$$

$$
B\,C^{-1}\,\hat{n}^{B} = \begin{bmatrix} \{(C_\beta S_\gamma - S_\beta S_\alpha C_\gamma)(-S_\sigma) + (-S_\beta C_\alpha)(C_\sigma)\} \\ \{(-S_\beta S_\gamma - C_\beta S_\alpha C_\gamma)(-S_\sigma) + (-C_\beta C_\alpha)(C_\sigma)\} \\ \{(-C_\alpha C_\gamma)(-S_\sigma) + (S_\alpha)(C_\sigma)\} \end{bmatrix} \tag{B-44}
$$

Hence, if we denote $\hat{n}^{A}$ by

$$
\hat{n}^{A} = n_x^{A}\,\hat{x}^{A} + n_y^{A}\,\hat{y}^{A} + n_z^{A}\,\hat{z}^{A} \tag{B-45}
$$

it follows from Equations (B-31), (B-33), and (B-34) that

$$
\begin{aligned}
n_x^{A} &= (C_\beta S_\gamma - S_\beta S_\alpha C_\gamma)(-S_\sigma) + (-S_\beta C_\alpha)(C_\sigma) \\
n_y^{A} &= (-S_\beta S_\gamma - C_\beta S_\alpha C_\gamma)(-S_\sigma) + (-C_\beta C_\alpha)(C_\sigma) \\
n_z^{A} &= (-C_\alpha C_\gamma)(-S_\sigma) + (S_\alpha)(C_\sigma)
\end{aligned} \tag{B-46}
$$

It is obvious from Figure B-5 that by definition

$$
n_y^{A} < 0 \tag{B-47}
$$

Next, let (x^{A}, y^{A}, z^{A}) denote the coordinates of the point Q in the auxiliary orbital system. It is noted that the coordinates of P are $(O, R + h, O)$. Hence, it follows that the following ratios are equal

$$
\frac{x^{A}}{n_x^{A}} = \frac{y^{A} - (R + h)}{n_y^{A}} = \frac{z^{A}}{n_z^{A}} \tag{B-48}
$$

which comprise a system of two independent equations and three unknowns. The third equation is obtained by noting that the point Q lies on the sphere

$$x^{A^2} + y^{A^2} + z^{A^2} = R^2 \tag{B-49}$$

Hence, we obtain

$$x^A = \frac{n_x^A}{n_z^A} z^A \tag{B-50}$$

$$y^A = \frac{n_y^A}{n_z^A} z^A + (R + h) \tag{B-51}$$

$$\left(\frac{n_x^A}{n_z^A}\right)^2 z^{A^2} + \left[\frac{n_y^A}{n_z^A} z^A + (R + h)\right]^2 + z^{A^2} = R^2$$

or, equivalently,

$$\left[\left(\frac{n_x^A}{n_z^A}\right)^2 + \left(\frac{n_y^A}{n_z^A}\right)^2 + 1\right]\frac{z^{A^2}}{R^2} + \left[2\,\frac{(R + h)}{R}\,\frac{n_y^A}{n_z^A}\right]\frac{z^A}{R} + \left[\frac{(R + h)^2}{R^2} - 1\right] = 0 \tag{B-52}$$

It is easily verified that Equation (B-52) reduces to Equation (B-7) for the case of zero pitch, roll, and yaw. Solution of Equation (B-52) yields

$$\frac{z^A}{R} = \frac{-B \pm \sqrt{(B^2 - 4AC)}}{2A} \tag{B-53}$$

$$A = \left[\left(\frac{n_x^A}{n_z^A}\right)^2 + \left(\frac{n_y^A}{n_z^A}\right)^2 + 1\right]$$

$$B = \left[2\,\frac{(R + h)}{R}\,\frac{n_y^A}{n_z^A}\right] \tag{B-54}$$

$$C = \left[\frac{(R + h)^2}{R^2} - 1\right]$$

Since the point of intersection Q is on the closer side to P, its z^A-coordinate is given by Equation (B-53) in which only the minus (-) sign is retained. Equations (B-50) and (B-41) are then used to yield x^A and y^A.

The rest of the analysis is precisely the same as that in Section B.1 commencing with Figure B-2. Thus, the coordinates (x^I, y^I, z^I) of Q in the inertial system and (x^R, y^R, z^R) in the rotating system are readily obtained by the use of Equations (B-22) through (B-24) and (B-25) through (B-27).

APPENDIX C - USE OF MASS STORAGE DEVICES AND THREE-DIMENSIONAL ARRAYS

The discussion of data bases in this report has assumed that only current data would be maintained in direct-access storage. A new class of storage devices, rapid mass stores, are now becoming available. These devices open up the possibility for on-line access to noncurrent data.

These devices are characterized by the ability to store on-line up to trillions (10^{12}) of bits of information, and to transfer large blocks of data at high rates exceeding 10^6 bits/seconds. A wide variety of mechanical and electro/optical devices with these characteristics is presently coming onto the market. These devices offer either write-once/multiple-read or multiple-write and read capabilities.

A write-once system would be suitable for archiving meteorological data bases for later reference, and for processing of temporal variations. Data can be transferred to the write-once device in the case of data bases with a random access device. There are four ways of archiving the data:

1. Storing raw data for later insertion into data base format
2. Storing a snap dump of the data base on completion of a satellite orbit
3. Storing a snap dump on a sparse schedule
4. Storing data in a three-dimensional data base (including a temporal coordinate)

This last method may have excellent long-range utility in that data would be blocked together in the temporal as well as spatial sense. This permits rapid access to temporally varying data in small spatial regions without long-range searches in the mass store. This is important since all mass storage devices utilize some sort of mechanical transport for units of the storage medium.

A three-dimensional data base can be constructed by extending the serial stringing scheme to 3 bits per level, utilizing 1 bit for each coordinate. A non-cubic configuration (with fewer temporal than spatial intervals) can be achieved by adding the third bit only to N levels of division, the number of temporal intervals being 2^N. The natural temporal unit would be one orbit which is approximately 100 minutes. The third bit could be added either at the highest or lowest level, depending on the application, for efficient access to areas selected by temporal or spatial order.

A device with 10^{12} bits would have the capacity for storage of more than 100 complete 10^8 byte data bases, and additional intervals for usage on selected spatial areas.

One mass storage device familiar to CSC personnel is the UNICON device used in the ILLIAC IV system. This device gives on-line access to 2.9×10^{12} bits. The storage medium is mylar strips which are written by burning areas with a focused laser beam. This device thus belongs to the write-once class.

Mechanically, the UNICON resembles the IBM 2301 Data Cell device, utilizing a rotating read/write drum on which the storage strip is positioned. Automatic mechanical pickers give on-line access to 1000 storage strips, each containing 2.9×10^9 bits of data. The data are spread over approximately 10^4 tracks/strip with a density of about 2.5×10^5 bits/track. Data can be transferred in 10^6-bit blocks at a rate of 4×10^6 bits/second. The interblock access time is 200 msec.

The general I/O management scheme described in Section 5 would still apply to such a device but data could be transferred in much longer records.

APPENDIX D - USE OF DISTRIBUTED PROCESSING

Implicit in Section 6 was the idea that the data base system would be implemented on a classic large-scale computer system. Substantial savings in central processor usage and greater flexibility could be obtained by distributing a number of system functions into various subsidiary devices.

A straight large-scale computer implementation would appear as shown in Figure D-1.

A simple improvement would be to place all I/O management under the control of a minicomputer which would handle all data base accesses and transfer data to the CPU via a DMA, extended core, or a staging drum as illustrated in Figure D-2.

A further elaboration would involve more elaborate minicomputer systems which would handle data input and user interaction functions. Figure D-3 shows an arrangement where a system of minicomputers relieves the CPU of all operations except calculations based on data transmitted from the data base. In such a system, either the CPU or the control minicomputer could control the system. Functions indicated as being performed on one minicomputer (such as input processing and mapping to the data base system) could actually be spread among a number of processors.

In the input processor area, orbit determination and attitude determination functions could take place in parallel in two or more units. In the mapping process, the following areas are amenable to parallel calculation:

- Mapping the ξ , η systems to x , y may be done in parallel $\xi \rightarrow x$ and $\eta \rightarrow y$
- Any number of interpolation blocks or scan lines could be processed in parallel if serial coordinates were outputted as strings to a separate I/O processor

- I/O processing for any number of spectral bands could be carried on in parallel

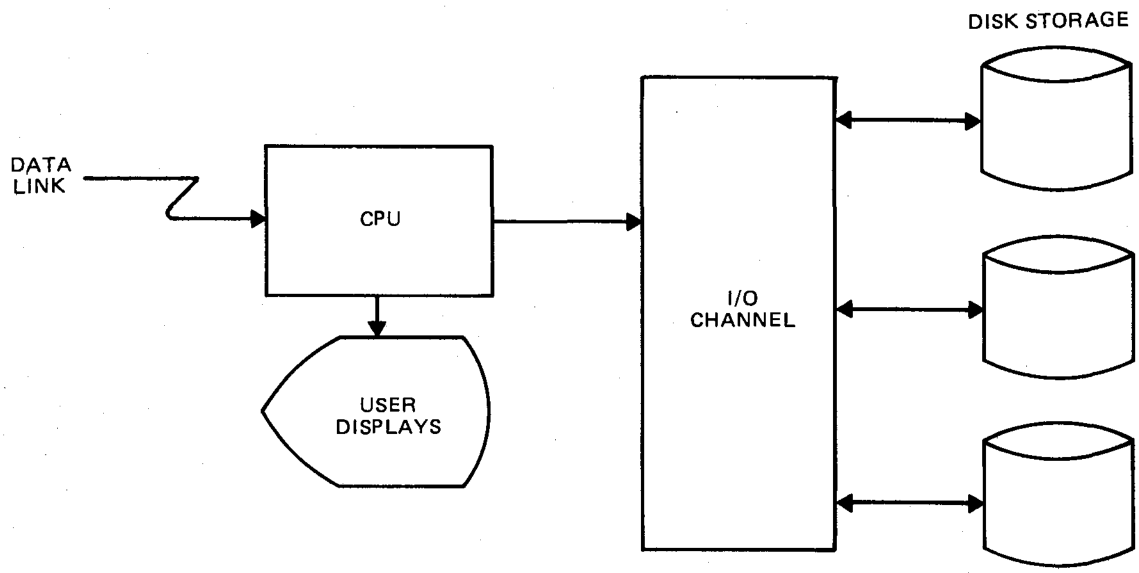

Figure D-1. Straight Large-Scale Computer Implementation

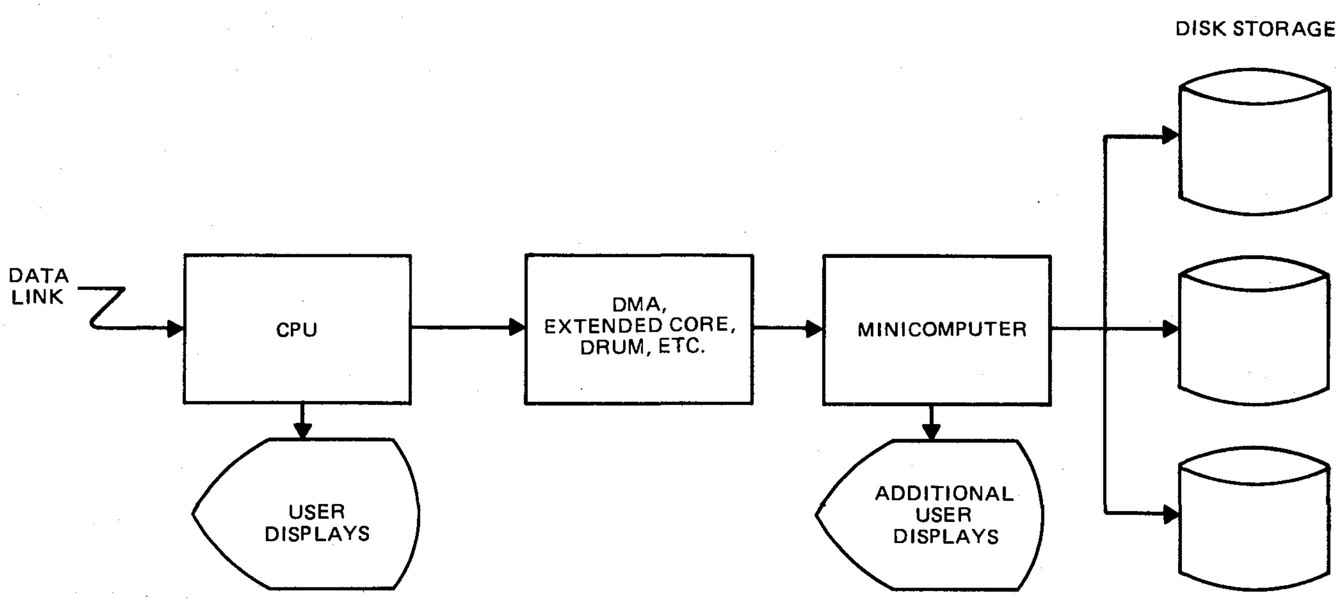

Figure D-2. Large-Scale Computer Implementation With I/O Management Under Minicomputer Control

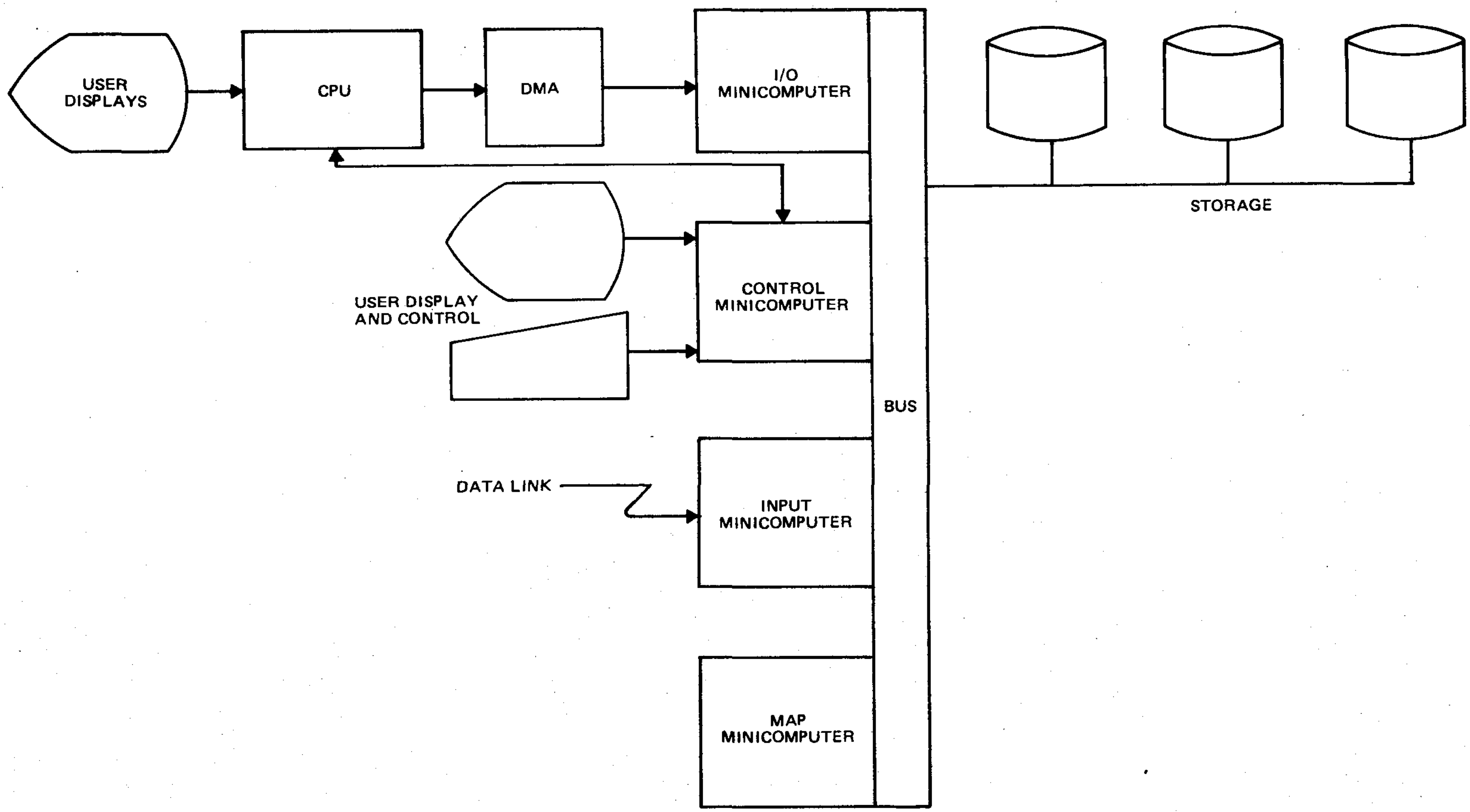

Figure D-3. Large-Scale Computer Implementation Where Either CPU or Minicomputer Controls System

APPENDIX E - LISTINGS OF TEST AND EVALUATION PROGRAMS

E.1 SUMMARY

This appendix contains listings of FORTRAN programs and subroutines used for testing and evaluating the various transformations and procedures covered in this report.

Listing	Section
Program used for evaluating coefficients of $f(x, y)$	E.2
Program used for evaluating coefficients of $f^*(\xi, \eta)$	E.3
Minimizer subroutine used by the programs of Sections E.2 and E.3	E.4
Subroutine used for generating a great circle arc	E.5
Subroutines used for mapping a point in Cartesian space into data base coordinates	E.6
Subroutines used for simulating the fast-filling method	E.7
Subroutines used for testing I/O handling method	E.8
Subroutines used for retrieving data from the data base and for graphic display	E.9

E.2 PROGRAM USED FOR EVALUATING COEFFICIENTS OF f(x , y)

```
C       MAIN
C
C       THIS PROGRAM COMPUTES COEFFICIENTS OF THE DIRECT TRANSFORMATION
C       FUNCTION BY MINIMIZING THE DIFFERENCE IN AREA FOR SMALL
C       AREAS
C
C       PROGRAMMER - E. MICHAEL O'NEILL
C                    COMPUTER SCIENCES CORPORATION
C                    8728 COLESVILLE RD. SILVER SPRING, MD.
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL FUNK
      COMMON RM,RO,GAMMA,OMGAM,XM,YM,XIN,YIN,DX,DY,DELTA,OMEGA
      COMMON /BILBO/  CHISQ,X(20),XMIN(20),XMAX(20),DELTAX(20),
     & DELMIN(20),ERR(20,20),NV,NTRACE,MATRIX,MASK(20)
      COMMON /FRODO/ NFMAX,NFLAT,JVARY,NFXTRA
      DATA PI/3.14159265358 9D0/
      DATA NOUT2/10/
C       SET PRECALCULATED COEFFICIENTS
      DELTA=0.7904864491208D0
      GAMSQ=PI/6.0D0
      GAMMA=DSQRT(GAMSQ)
      OMGAM=1.0D0-GAMMA
      AMU=DSQRT(DSQRT(3.0D0)*PI/2.0D0)
C       SET DIMENSIONAL PARAMETERS
      RM=1.0D0
      RO=DSQRT(RM*RM/3.0D0)
      R2=RO*RO
      R4=R2*R2
      OMEGA=0.5D0/R4*(3.0D0-2.0D0*GAMMA-AMU-2.0D0*DELTA*R4)
      XM=RO
      YM=RO
      XIN=XM/  5.0D0
      YIN=XIN
      DX=XM*1.0D-12
      DY=DX
      WRITE(NOUT2,1001)
 1001 FORMAT(' INITIAL CONDITION OUTPUT,'
     1'ALL VARIABLE PARAMETERS ZEROED'/)
C       COMPUTE AND DISPLAY INITIAL AREAS
      CALL DISPSQ(6)
      CALL DISPSQ(NOUT2)
  101 CONTINUE
C       READ INITIAL VALUES AND LIMITS FROM CARDS
      READ(5,1,END=99) NV,NFMAX,NTRACE,MATRIX,MASK
      WRITE(6,2)NV, NFMAX, NTRACE, MATRIX,MASK
    1 FORMAT(4I4,4X,20I1)
    2 FORMAT(' NV, NFMAX, NTRACE, MATRIX, MASK ',4I4,4X,20I1)
      READ(5,5) (X(I),I=1,NV)
      WRITE(6,6)(X(I),I=1,NV)
      READ(5,5) (XMAX(I),I=1,NV)
      WRITE(6,6)(XMAX(I),I=1,NV)
      READ(5,5) (XMIN(I),I=1,NV)
      WRITE(6,6)(XMIN(I),I=1,NV)
      READ(5,5) (DELTAX(I),I=1,NV)
      WRITE(6,6)(DELTAX(I),I=1,NV)
      READ(5,5) (DELMIN(I),I=1,NV)
      WRITE(6,6)(DELMIN(I),I=1,NV)
    5 FORMAT(10F8.3)
    6 FORMAT(1X,10F8.3)
      CALL DISPSQ(6)
C
C       CALL STEPIT TO MINIMIZE RMS AREA DEVIATIONS AS CALCULATED
C       BY SUBROUTINE FUNK
C
      CALL STEPIT(FUNK)
C       DISPLAY RESULTS
      CALL DISPSQ(6)
      WRITE(NOUT2,1002) CHISQ,(X(I),I=1,NV)
 1002 FORMAT(' CHISQ,VARIABLE PARAMETERS ', D12.5/(3D21.14/))
      CALL DISPSQ(NOUT2)
      GO TO 101
   99 CONTINUE
      STOP
      END
      BLOCK DATA
C       INITIALIZE STEPIT COMMON AREAS
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BILBO/  CHISQ,X(20),XMIN(20),XMAX(20),DELTAX(20),
     & DELMIN(20),ERR(20,20),NV,NTRACE,MATRIX,MASK(20)
      DATA X/20*0.0D0/, XMAX/20*0.1D-10/, XMIN/20*-0.1D-10/,
     *      DELTAX/20*0.1D-11/, DELMIN/20*0.1D-13/
      END
```

```
      SUBROUTINE DISPSQ(NOUT)
C
C     THIS SUBROUTINE PRINTS OUT THE ABSOLUTE AND RELATIVE AREAS
C     OF A GRID OF SMALL SQUARES IN ONE OCTANT
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SA(6,6), AA(6,6)
      DATA SA,AA/36*0.0D0,36*0.0D0/
      CALL CNTARY(AA)
      SAC=AA(1,1)
      DO 30 J=1,6
      DO 30 I=1,J
   30 SA(I,J)=AA(I,J)/SAC
      WRITE(NOUT,40) ((AA(I,J),I=1,J),J=1,6)
      WRITE(NOUT,45)
      WRITE(NOUT,40) ((SA(I,J),I=1,J),J=1,6)
      WRITE(NOUT,45)
      WRITE(NOUT,45)
   40 FORMAT(1X,D12.5/1X,2D12.5/1X,3D12.5/1X,4D12.5/1X,5D12.5/
     1 1X,6D12.5)
   45 FORMAT(1H )
      RETURN
      END
      SUBROUTINE FUNK
C
C     THIS SUBROUTINE CALLS CNTARY TO CALCULATE THE PROJECTED
C     AREAS OF A GRID OF SMALL SQUARES.
C     THE RMS OF THE DEVIATIONS OF THE AREAS FROM THE AREA CALCULATED
C     AT THE CENTER IS RETURNED AS CHISQ
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON RM,RO,GAMMA,OMGAM,XM,YM,XIN,YIN,DX,DY
      COMMON /BILBO/  CHISQ,X(20),XMIN(20),XMAX(20),DELTAX(20),
     & DELMIN(20),ERR(20,20),NV,NTRACE,MATRIX,MASK(20)
      DIMENSION SA(6,6), AA(6,6)
      EQUIVALENCE (C01,X(1)),(C10,X(1)),(C11,X(2)),(C02,X(3)),(C20,X(3))
     *,(C12,X(4)),(C21,X(4)),(C22,X(5)),(C03,X(6)),(C30,X(6)),(C31,X(7))
     *,(C13,X(7)),(C23,X(8)),(C32,X(8)),(C33,X(9))
      CALL CNTARY(SA)
      SAC=SA(1,1)
      DO 20 J=1,6
      DO 20 I=1,J
   20 SA(I,J)=SA(I,J)/SAC
      DEVSQ=0.0D0
      DO 30 J=1,6
      DO 30 I=1,J
   30 DEVSQ=DEVSQ+(SA(I,J)-1.0D0)**2
      CHISQ=DSQRT(DEVSQ/ 21.0D0)
      RETURN
      END
```

```
      SUBROUTINE CNTARY(SA)
C
C      THIS SUBROUTINE CALCULATES THE PROJECTED AREAS OF A GRID OF
C      SMALL SQUARES BY FIRST APPLYING THE DIRECT TRANSFORMATION
C      TO A SET OF POINTS, COMPUTING THE AREA OF THE TRANSFORMED SQUARE
C      AND PROJECTING TO THE SURFACE OF THE SPHERE
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON RM,RO,GAMMA,OMGAM,XM,YM,XIN,YIN,DX,DY
      DIMENSION P1(2), P2(2), P3(2), P4(2), SA(6,6)
C
C      SPHERICAL PROJECTION FUNCTION
      SPROJ(A,B,C,D,E)=A*(B*B*C)/((C*C+D*D+E*E)**1.50D0)
C
      X=0.0D0
      DO 10 J=1,5
      Y=0.0D0
      DO 5 I=1,J
C      DIRECT TRANSFORMATION
      CALL XY TO PE (X,Y,RO,P1(1),P1(2))
      CALL XY TO PE (X+DX,Y    ,RO,P2(1),P2(2))
      CALL XY TO PE (X+DX,Y+DY,RO,P3(1),P3(2))
      CALL XY TO PE (X    ,Y+DY,RO,P4(1),P4(2))
C      COMPUTE AREA OF TRANSFORMED SQUARE
      S1=AREA(P1,P2,P3)
      S2=AREA(P1,P4,P3)
      S=S1+S2
C      PROJECT TO SPHERICAL SURFACE
      SA(I,J)=SPROJ(S,RM,RO,P1(1),P1(2))
    5 Y=Y+YIN
   10 X=X+XIN
      X=0.0D0
      Y=YM
      DO 20 I=1,6
      CALL XY TO PE (X    ,Y    ,RO,P1(1),P1(2))
      CALL XY TO PE (X-DX,Y    ,RO,P2(1),P2(2))
      CALL XY TO PE (X-DX,Y-DY,RO,P3(1),P3(2))
      CALL XY TO PE (X    ,Y-DY,RO,P4(1),P4(2))
      S      =AREA(P1,P2,P3)+AREA(P1,P4,P3)
      SA(I, 6)=SPROJ(S,RM,RO,P1(1),P1(2))
      X=X+XIN
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE XY TO PE (XI,YI,R,P,E)
C      DIRECT TRANSFORMATION FUNCTIONS
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON RM,RO,GAMMA,OMGAM,XM,YM,XIN,YIN,DX,DY,DELTA,OMEGA
      COMMON /BILBO/  CHISQ,X(20),XMIN(20),XMAX(20),DELTAX(20),
     & DELMIN(20),ERR(20,20),NV,NTRACE,MATRIX,MASK(20)
      EQUIVALENCE (C00,X(1)),(C10,X(2)),(C01,X(3)),
     1            (C11,X(4)),(C20,X(5)),(C02,X(6)),
     2            (D00,X(7)),(D01,X(8)),(D02,X(9))
      R2=R*R
      R4=R2*R2
      XSQ=XI*XI
      X3=XSQ*XI
      X4=XSQ*XSQ
      YSQ=YI*YI
      Y3=YSQ*YI
      Y4=YSQ*YSQ
      P=GAMMA*XI+(OMGAM/R2)*X3+
     1 XI*YSQ*(R2-XSQ)*(DELTA+(R2-YSQ)*(C00+C10*XSQ+C01*YSQ+
     2 C11*XSQ*YSQ+C20*X4+C02*Y4))+
     3 X3*(R2-XSQ)*(OMEGA+(R2-XSQ)*(D00+D01*XSQ+D02*X4))
      E=GAMMA*YI+(OMGAM/R2)*Y3+
     1 YI*XSQ*(R2-YSQ)*(DELTA+(R2-XSQ)*(C00+C10*YSQ+C01*XSQ+
     2 C11*YSQ*XSQ+C20*Y4+C02*X4))+
     3 Y3*(R2-YSQ)*(OMEGA+(R2-YSQ)*(D00+D01*YSQ+D02*Y4))
      RETURN
      END
C      CALCULATE THE AREA OF A TRIANGLE FORMED BY THREE CARTESIAN POINTS
      FUNCTION AREA(P1,P2,P3)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P1(2),P2(2),P3(2)
      ASQ=(P1(1)-P2(1))**2+(P1(2)-P2(2))**2
      BSQ=(P2(1)-P3(1))**2+(P2(2)-P3(2))**2
      CSQ=(P3(1)-P1(1))**2+(P3(2)-P1(2))**2
      B=DSQRT(BSQ)
      C=DSQRT(CSQ)
      COSA=-(ASQ-BSQ-CSQ)/(2.0D0*B*C)
      CP=B*COSA
      H=BSQ-CP**2
      IF(H.LE.0.0D0) GO TO 100
      H=DSQRT(H)
       AREA=0.5D0*C*H
      RETURN
  100 AREA=0.0D0
      RETURN
      END
```

E.3 PROGRAM USED FOR EVALUATING COEFFICIENTS OF $f^*(\xi, \eta)$

```
C        MAIN
C
C        THIS PROGRAM COMPUTES THE COEFFICIENTS OF THE INVERSE
C        TRANSFORMATION FUNCTION BY MINIMIZING THE DEVIATIONS OF
C        A RETRANSFORMED GRID
C
C        PROGRAMMER - E. MICHAEL O'NEILL
C                     COMPUTER SCIENCES CORPORATION
C                     8728 COLESVILLE RD. SILVER SPRING, MD.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL FUNK
      COMMON RS,R,RSQ,R4,XIN,YIN
      COMMON/FRODO/ NFMAX,NFLAT,JVARY,NEXTRA
      COMMON /BILBO/ DIFF,VAR(20),VMIN(20),VMAX(20),DV(20),
     1              DELMIN(20),ERR(20,20),NV,NT,MAT,MASK(20)
C       SET DIMENSIONAL CONSTANTS
      RS=1.0D0
      R=0.577350269189600
      RSQ=0.33333333333333D0
      R4=0.11111111111111D0
C       INITIALIZE COEFFICIENTS
      DO 10 I=1,20
      VAR(I)=0.0D0
      VMIN(I)=-0.5D3
      VMAX(I)= 0.5D3
      DV(I)=0.1D0
      DELMIN(I)=0.1D-10
   10 MASK(I)=0
C       NUMBER OF COEFFICIENTS
      NV=13
C       NUMBER OF ITERATIONS
      NFMAX=2000
      NT=0
      MAT=0
      XIN=R/5.0D0
      YIN=XIN
C
C         STEPIT CALLS SUBROUTINE FUNK WHICH CALCULATES THE RMS
C         DEVIATION OF THE RETRANSFORMED GRID, STEPIT ITERATES UNTIL
C         A SATISFACTORY RMS HAS BEEN REACHED OR UNTIL NFMAX ITERATIONS
C
       CALL STEPIT(FUNK)
C
C         USE FINAL COEFFICIENTS TO CALCULATE RETRANSFORMED GRID AND PRINT
C
       X=0.0D0
       DO 20 IX=1,6
       Y=0.0D0
       DO 15 IY=1,IX
C        DIRECT TRANSFORMATION
       P=FDMAP(X,Y)
       E=FDMAP(Y,X)
C        INDIRECT TRANSFORMATION
       XB=FIMAP(P,E)
       YB=FIMAP(E,P)
       DIFF=(X-XB)**2+(Y-YB)**2
       WRITE(6,12) X,Y,P,E,XB,YB,DIFF
       WRITE(1,12) X,Y,P,E,XB,YB,DIFF
   12  FORMAT(' X,Y,P,E,XB,YB,DIFF ',7F8.5)
   15  Y=Y+YIN
   20  X=X+XIN
       STOP
       END
```

```
      SUBROUTINE FUNK
C
C      THIS SUBROUTINE CALCULATES THE SUM OF THE SQUARES OF THE
C      DEVIATIONS OF THE RETRANSFORMED GRID
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/BILBO/ DIFF,VAR(20)
      COMMON RS,R,RSQ,R4,XIN,YIN
      DIFF=0.0D0
      X=0.0D0
      DO 10 IX=1,6
      Y=0.0D0
      DO 5 IY=1,IX
C      DIRECT TRANSFORMATION
      P=FDMAP(X,Y)
      E=FDMAP(Y,X)
C       INVERSE TRANSFORMATION
      XB=FIMAP(P,E)
      YB=FIMAP(E,P)
      DIFF=DIFF+(X-XB)**2+(Y-YB)**2
    5 Y=Y+YIN
   10 X=X+XIN
      RETURN
      END
```

```
      FUNCTION FDMAP(A,B)                                               00005500
C
C      THIS FUNCTION EVALUATES THE DIRECT TRANSFORMATION AT A,B USING
C      A PREDETERMINED SET OF COEFFICIENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)                                         00005600
      COMMON RS,R,RSQ,R4,XIN,YIN                                        00005700
      DATA GAMMA,OMGAM/0.7236012545582D0,0.276398745441 8D0/            00005710
      DATA DELTA,OMEGA/0.790486449120 8D0,-0.122544148798 4D1/          00005720
      DATA C00,C10,C01/-0.272170536D1,-0.558421683D1,0.217111748D1/     00005730
      DATA C11,C20,C02/-0.641601515D1,-0.345786274D1,0.197362657D1/     00005740
      DATA D0,D1,D2/0.148331292D1,0.111997260D1,0.605153821D1/          00005750
      ASQ=A*A                                                           00005800
      A3=A*ASQ                                                          00005900
      A4=ASQ*ASQ                                                        00006000
      BSQ=B*B                                                           00006100
      B4=BSQ*BSQ                                                        00006200
      RMA=RSQ-ASQ                                                       00006300
      FDMAP=GAMMA*A+OMGAM/RSQ*A3+                                       00006400
     1       A3*RMA*(OMEGA+RMA*(D0+D1*ASQ+D2*A4))+                      00006500
     2       A*BSQ*RMA*(DELTA+(RSQ-BSQ)*                                00006600
     3       (C00+C10*ASQ+C01*BSQ+C11*ASQ*BSQ+C20*A4+C02*B4))           00006700
      RETURN                                                            00006800
       END                                                              00006900
      FUNCTION FIMAP(A,B)                                               00004200
C
C      THIS FUNCTION EVALUATES THE INVERSE TRANSFORMATION USING A
C      SET OF COEFFICIENTS INPUTTED THROUGH COMMON/BILBO/
C
      IMPLICIT REAL *8 (A-H,O-Z)                                        00004300
      COMMON RS,R,RSQ,R4,XIN,YIN                                        00004400
      COMMON/BILBO/ DIF,VAR(20)                                         00004500
      EQUIVALENCE (DELTA,VAR(1)),(OMEGA,VAR(2)),                        00004525
     1          (DELT1,VAR(3)),(OMEG1,VAR(4)),                          00004550
     1           (C00,VAR(5)),(C10,VAR(6)),(C01,VAR(7)),                00004600
     1           (C11,VAR(8)),(C20,VAR(9)),(C02,VAR(10)),               00004650
     1           (D0,VAR(11)),(D1,VAR(12)),(D2,VAR(13))                 00004700
      DATA GAMMA,OMGAM/0.1381976597886D1,-0.3819765978855D0/            00004805
      ASQ=A*A                                                           00004810
      A3=A*ASQ                                                          00004815
      A4=ASQ*ASQ                                                        00004820
      BSQ=B*B                                                           00004825
      B4=BSQ*BSQ                                                        00004830
      RMA=RSQ-ASQ                                                       00004835
      RMA=RSQ-ASQ                                                       00004900
      FIMAP=GAMMA*A+OMGAM/RSQ*A3+                                       00005000
     1      A3*RMA*(OMEGA+OMEG1*(RSQ-BSQ)+RMA*(D0+D1*ASQ+D2*A4))+       00005100
     2      A*BSQ*RMA*(DELTA+DELT1*RMA+(RSQ-BSQ)*                       00005150
     3      (C00+C10*ASQ+C01*BSQ+C11*ASQ*BSQ+C20*A4+C02*B4))            00005200
      RETURN                                                            00005300
      END                                                               00005400
```

E.4 MINIMIZER SUBROUTINE USED BY THE PROGRAMS OF SECTIONS E.2 AND E.3

```
      SUBROUTINE STEPIT %FUNK<                                          GRH03140
      IMPLICIT REAL*8 (A-H,O-Z)                                         GRH03150
C                                                                       GRH03160
C                                                                       GRH03180
C  360 STEPIT, MODIFIED BY J. BENSINGER, U. PENNSYLVANIA, 1971
C                                                                       GRH031A0
C  STEPIT FINDS LOCAL MINIMA OF A SMOOTH FUNCTION OF SEVERAL PARAMETERS.GRH031B0
C                                                                       GRH031C0
C  @STEPIT IS A PHLEGMATIC METHOD OF SOLVING A PROBLEM.@                GRH031D0
C           --  J. H. BURRILL, JR., @360 STEPIT - A USER@S MANUAL@      GRH031E0
C                                                                       GRH031F0
C  THIS SOURCE DECK AND A LONG WRITE-UP ARE AVAILABLE FROM ....         GRH03200
C                                                                       GRH03210
C      QUANTUM CHEMISTRY PROGRAM EXCHANGE                               GRH03220
C      DEPT. OF CHEMISTRY, INDIANA UNIVERSITY                           GRH03230
C      BLOOMINGTON, INDIANA 47401                                       GRH03240
C                                                                       GRH03250
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *GRH03260
C                                                                       GRH03270
C     FUNK         --  THE NAME OF THE SUBROUTINE THAT COMPUTES CHISQ   GRH03280
C                             GIVEN X%1<,X%2<,...,X%NV<                 GRH03290
C     NV           --  THE NUMBER OF PARAMETERS, X                      GRH032A0
C     NTRACE       --  #0 FOR NORMAL OUTPUT, #&1 FOR TRACE OUTPUT,      GRH032B0
C                             #-1 FOR NO OUTPUT                         GRH032C0
C     MATRIX       --  #0 FOR NO ERROR CALCULATION, #100&M FOR ERROR    GRH032D0
C                             CALCULATION USING STEPS 10**M TIMES LARGERGRH032E0
C                             THAN THE LAST STEPS USED IN THE MINIMIZATION GRH032F0
C     CHISQ        --  THE VALUE OF THE FUNCTION TO BE MINIMIZED        GRH03300
C     MASK%J<      --  NONZERO IF X%J< IS TO BE HELD FIXED              GRH03310
C     X%J<         --  THE J-TH PARAMETER                               GRH03320
C     XMAX%J<      --  THE UPPER LIMIT ON X%J<                          GRH03330
C     XMIN%J<      --  THE LOWER LIMIT ON X%J<                          GRH03340
C     DELTAX%J<    --  THE INITIAL STEP SIZE FOR X%J<                   GRH03350
C     DELMIN%J<    --  THE LOWER LIMIT %CONVERGENCE TOLERANCE< ON THE   GRH03360
C                             STEP SIZE FOR X%J<                        GRH03370
C     ERR%J,K<     --  RETURNS THE ERROR MATRIX %ALSO USED FOR SCRATCH  GRH03380
C                             STORAGE<                                  GRH03390
C     NFMAX        --  THE MAXIMUM NUMBER OF FUNCTION COMPUTATIONS      GRH033A0
C     NFLAT        --  NONZERO IF THE SEARCH IS TO TERMINATE WHEN ALL   GRH033B0
C                             TRIAL STEPS GIVE IDENTICAL FUNCTION VALUESGRH033C0
C     JVARY        --  STEPIT SETS JVARY NONZERO IF X%JVARY< IS THE ONLYGRH033D0
C                             X%J< THAT HAS CHANGED SINCE THE LAST CALL TO GRH033E0
C                             FUNK                                      GRH033F0
C     NEXTRA       --  USED BY SUBROUTINE SIMPLEX BUT NOT BY STEPIT     GRH03400
C                                                                       GRH03410
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *GRH03420
C                                                                       GRH03430
C  THE DIMENSIONS OF ALL VECTORS AND MATRICES ARE NV, EXCEPT FOR ....   GRH03440
C     ERR%NV,MAX%NV,MOSQUE<<,SECOND%2,2<,XOSC%NV,MOSQUE<,CHIOSC%MOSQUE<.GRH03450
C                                                                       GRH03460
C  TO REDUCE STORAGE TO A MINIMUM, SET MOSQUE#0, REDIMENSION ERR%1,1<,  GRH03470
C  XOSC%1,1<, SALVO%1<, AND CHIOSC%1<, AND DELETE THE OSCILLATION       GRH03480
C  SEARCH AND ERROR COMPUTATION SECTIONS %SEE COMMENT CARDS BELOW<.     GRH03490
C  OR, USE SUBROUTINE STP, WHICH HAS THESE DELETIONS PLUS DELETION OF   GRH034A0
C  THE COLINEARITY CHECK.                                               GRH034B0
C                                                                       GRH034C0
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *GRH034D0
C                                                                       GRH034E0
      COMMON /BILBO/  CHISQ,X(20),XMIN(20),XMAX(20),DELTAX(20),         GRH034F0
     & DELMIN(20),ERR(20,20),NV,NTRACE,MATRIX,MASK(20)                  GRH03500
C                                                                       GRH03510
      COMMON /FRODO/ NFMAX,NFLAT,JVARY,NEXTRA                           GRH03520
C                                                                       GRH03530
      DIMENSION VEC%20<,TRIAL%20<,XSAVE%20<,CHI%20<,DX%20<,SECOND%2,2<  GRH03540
      DIMENSION OLDVEC%20<,SALVO%20<,XOSC%20,15<,CHIOSC%15<,JFLAT%20<   GRH03550
C                                                                       GRH03560
C  TO RUN ON THE CDC 3600, THE FOLLOWING DATA STATEMENT MUST BE         GRH03570
C  ACTIVATED AND SUBROUTINE SSWTCH MUST BE SUPPLIED.                    GRH03580
C  TO RUN ON OTHER COMPUTERS, A BLOCK DATA SUBPROGRAM MUST BE SUPPLIED  GRH03590
C  TO SET NFMAX AND NFLAT.                                              GRH035A0
C                                                                       GRH035B0
C     DATA %NFMAX#1000000<,%NFLAT#1<                                    GRH035C0
C                                                                       GRH035D0
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *GRH035E0
C                                                                       GRH035F0
C  SET FIXED QUANTITIES ....                                            GRH03600
C                                                                       GRH03610
C                               KW ...  STANDARD OUTPUT UNIT NUMBER     GRH03620
C                                                                       GRH03630
C                                  IBM 709/7090/7094 ....               GRH03640
      KW#6                                                              GRH03650
C                                  CDC 3600/6400 ....                   GRH03660
C     KW#61                                                             GRH03670
C                               KTYPE ... CONSOLE TYPEWRITER UNIT       GRH03680
```

```
C
C                                   CDC 3600 ....
      KTYPE#6
C
C                          SIGNIF ...  LARGEST NUMBER SUCH THAT
C                                %SIGNIF&1.<-SIGNIF IS NONZERO
C
C                                IBM 709/7090/7094 ....
      SIGNIF#2.E8
C                                IBM 360
C
C       DOUBLE PRECISION
      SIGNIF#1.D16
C                                CDC 3600 ....
      SIGNIF#1.E10
C                                CDC 6400 ....
      SIGNIF#1.E14
C
C                          HUGE ...  THE LARGEST FLOATING POINT
C                                NUMBER
C
C                                IBM 709/7090/7094 ....
      HUGE#1.E37
C                                IBM 360
      HUGE#1.D75
C                                CDC 3600/6400 ....
      HUGE#1.E307
C
C                          NVMAX ...  MAXIMUM VALUE OF NV
      NVMAX#20
C                          MOSQUE ...  MAXIMUM DEPTH OF OSCILLATION
C                                SEARCH
      MOSQUE#15
C                          RATIO ...  RATIO OF STEP SIZE DECREASE
      RATIO#10.0
C                          COLIN ...  COLINEARITY TOLERANCE
      COLIN#0.99
C                          NCOMP ...  MAXIMUM NUMBER OF CYCLES
C                                WITHOUT ATTEMPTING A GIANT STEP
      NCOMP#5
C                          ACK ...  RATIO OF STEP SIZE INCREASE
      ACK#2.0
C                          FACUP ...  IF MORE THAN FACUP STEPS ARE
C                                TAKEN, THE STEP SIZE IS INCREASED
      FACUP#4.
C                          NSTPMX ...  LOG2%MAXIMUM NUMBER OF STEPS<
      NSTPMX#5
C                          NSSW ...  TERMINATION SENSE SWITCH NUMBER
      NSSW#6
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C   CHECK SOME INPUT QUANTITIES, AND SET THEM TO STANDARD VALUES IF
C   DESIRED.
C
C                          DECOMMENT FOR SSW OPTION ....
C     CALL SSWTCH %NSSW,JUMP<
C     IF%JUMP-1<10,10,40
C                          ONLY USAGE OF THE CONSOLE TYPEWRITER....
C  10 WRITE %KTYPE,20< NSSW
C  20 FORMAT%23H TURN OFF SENSE SWITCH I2<
C  30 CALL SSWTCH %NSSW,JUMP<
C     IF%JUMP-1<30,30,40
C
   40 IF%NV<430,430,50
   50 IF%NCOMP<60,60,70
   60 NCOMP#1
   70 JVARY#0
C                          NACTIV ...  NUMBER OF ACTIVE X%J<
      NACTIV#0
      DO 230 I#1,NV
      IF%MASK%I<<230,80,230
   80 DELX#DELTAX%I<
      IF%DELX*X%I<<90,100,100
   90 DELX#-DELX
  100 XPLUS#X%I<&DELX
      IF%XPLUS-X%I<<140,110,140
  110 IF%X%I<<130,120,130
  120 DELTAX%I<#0.01
      GO TO 140
  130 DELTAX%I<#0.01*X%I<
  140 IF%DELMIN%I<<160,150,170
  150 DELMIN%I<#DELTAX%I</SIGNIF
      IF%DELMIN%I<<160,170,170
  160 DELMIN%I<#-DELMIN%I<
  170 IF%XMAX%I<-XMIN%I<<180,180,190
  180 XMAX%I<#HUGE
```

```
      XMIN%I<#-HUGE
  190 NACTIV#NACTIV&1
C                                 X%I<#AMAX1%XMIN%I<,AMIN1%XMAX%I<,X%I<<<
      IF%X%I<-XMAX%I<<210,210,200
  200 X%I<#XMAX%I<
  210 IF%X%I<-XMIN%I<<220,230,230
  220 X%I<#XMIN%I<
  230 CONTINUE
C
C     COMPUTE COMPAR.  THE PROBABILITY THAT THE COSINE OF THE ANGLE
C     BETWEEN TWO RANDOM DIRECTIONS EXCEEDS COMPAR IS APPROXIMATELY
C     %1-COLIN</2 .
C
      COMPAR#0.0
      IF%NACTIV-1<240,300,260
  240 DO 250 J#1,NV
  250 MASK%J<#0
      GO TO 50
  260 A#NACTIV
      SUB#2.0/%A-1.0<
C     P#2.0*%1.0/SQRT  %A</%1.0-0.5**SUB<-1.0<
      P#2.0*%1.0/DSQRT%A</%1.0-0.5**SUB<-1.0<
      COMPAR#%1.-%1.-COLIN<**SUB<*%1.&P*%1.-COLIN<<
C                                 COMPAR#MIN1F%.999,ABSF%COMPAR<<
      IF%COMPAR<270,280,280
  270 COMPAR#-COMPAR
  280 IF%COMPAR-.999<300,300,290
  290 COMPAR#.999
C
  300 IF%NTRACE<390,310,310
  310 WRITE%           KW,320<
  320 FORMAT%92H1ENTER SUBROUTINE STEPIT.  ORIGINAL VERSION BY  CHANDLER
     *, PHYSICS DEPT., INDIANA UNIVERSITY.//19H INITIAL VALUES..../<
      WRITE%           KW,330<%MASK%J<,J#1,NV<
  330 FORMAT%/10H MASK    #   9%I6,6X</%4X9I12<<
      WRITE%           KW,340<%X%J<,J#1,NV<
  340 FORMAT%/10H X       #   9E12.4/%10X 9E12.4<<
      WRITE%           KW,350<%XMAX%J<,J#1,NV<
  350 FORMAT%/10H XMAX    #   9E12.4/%10X 9E12.4<<
      WRITE%           KW,360<%XMIN%J<,J#1,NV<
  360 FORMAT%/10H XMIN    #   9E12.4/%10X 9E12.4<<
      WRITE%           KW,370<%DELTAX%J<,J#1,NV<
  370 FORMAT%/10H DELTAX  #   9E12.4/%10X 9E12.4<<
      WRITE%           KW,380<%DELMIN%J<,J#1,NV<
  380 FORMAT%/10H DELMIN  #   9E12.4/%10X 9E12.4<<
  390 CALL FUNK
      CHISAV#CHISQ
      CALL FUNK
C                                 NF ...  NUMBER OF FUNCTION CALLS
      NF#2
      IF%CHISQ-CHISAV<400,420,400
  400 WRITE %KW,410< CHISAV,CHISQ,NF
  410 FORMAT%///3%/59H WARNING....  CHISQ IS NOT A REPRODUCIBLE FUNCTION
     * OF X%J<.  </5X 9HCHISAV # E22.14,5X 8HCHISQ # E22.14,5X5HNF # I5<
C
C                                 JOCK ...  SWITCH USED IN SETTING JVARY
  420 JOCK#1
      IF%NTRACE<450,430,430
  430 WRITE%           KW,440<NV,NACTIV,MATRIX,NCOMP,NFMAX,NFLAT,
     *    RATIO,ACK,COLIN,COMPAR,CHISQ
  440 FORMAT%//1H I3,11H VARIABLES,I3,8H ACTIVE.10X8HMATRIX #I4,10X7HNCO
     *MP #I2,10X7HNFMAX #I8,10X7HNFLAT #I2//
     *            8H RATIO #F5.1,10X5HACK #F5.1,10X7HCOLIN #F6.3,10X8HCOMPAR
     * #F6.3///8H CHISQ #E15.8///23H BEGIN MINIMIZATION....  <
  450 IF%NV<2120,2120,460
  460 IF%NTRACE<490,490,470
  470 WRITE%           KW,480<
  480 FORMAT%//60%1X1H*<//10X37HTRACE MAP OF THE MINIMIZATION PROCESS//<
C
  490 DO 500 I#1,NV
C                                 VEC%J< ...  CURRENT VECTOR OF NUMBER OF
C                                      STEPS IN X%J<
      VEC%I<#0.
C                                 DX%J< ...  CURRENT STEP SIZE FOR X%J<
  500 DX%I<#DELTAX%I<
C                                 CHIOLD ...  BEST PREVIOUS VALUE OF CHISQ
      CHIOLD#CHISQ
C                                 NOSC ...  CURRENT DEPTH OF THE OSCILLATION
C                                      INFORMATION
      NOSC#0
C                                 KWIT ...  TERMINATION SWITCH
      KWIT#0
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C
C     VARY THE PARAMETERS ONE AT A TIME.
C     THIS IS THE STARTING POINT USED EACH TIME THE STEP SIZE IS REDUCED
C     OR A SUCCESSFUL GIANT STEP IS COMPLETED.
C
C                                     NCIRC ... NUMBER OF CONSECUTIVE X%J<
C                                         WITHOUT SIZEABLE CHANGES
  510 NCIRC#0
C                                     NZIP ... NUMBER OF CONSECUTIVE CYCLES
C                                         WITHOUT A GIANT STEP
      NZIP#0
C
C     MAIN DO LOOP FOR CYCLING THROUGH THE VARIABLES.
C     FIRST TRIAL STEP WITH EACH VARIABLE IS SEPARATE.
C
C                                     NACK ... NUMBER OF ACTIVE X%J< CYCLED
C                                         THROUGH
  520 NACK#0
      DO 1750 I#1,NV
C                                     JFLAT%J< ... NONZERO IF CHANGING X%J< DID
C                                         NOT CHANGE CHISQ
      JFLAT%I<#0
C                                     OLDVEC%J< ... OLD VECTOR OF NUMBER OF
C                                         STEPS IN X%J<
      OLDVEC%I<#VEC%I<
      VEC%I<#0.0
C                                     TRIAL%J< ...  CHANGE IN X%J<
      TRIAL%I<#0.0
      IF%MASK%I<<530,540,530
  530 VEC%I<#-0.0
      JFLAT%I<#1
      GO TO 1730
  540 NACK#NACK&1
      SAVE#X%I<
C                                     CHECK THAT DX%I< IS NOT NEGLIGIBLE.
      DELX#DX%I<
      IF%DELX*SAVE<550,560,560
  550 DELX#-DELX
  560 XPLUS#SAVE&DELX
      IF%XPLUS-SAVE<580,570,580
  570 JFLAT%I<#2
      GO TO 770
C                                     ADX#ABS%DX%I<<
  580 ADX#DX%I<
      IF%ADX<590,600,600
  590 ADX#-ADX
C                                     STEP X%I<.
  600 X%I<#SAVE&DX%I<
      JVARY#0
      IF%JOCK<620,620,610
  610 JOCK#0
      JVARY#I
  620 NFLAG#1
      IF%X%I<-XMIN%I<<640,630,630
  630 IF%X%I<-XMAX%I<<650,650,640
  640 NFLAG#NFLAG&3
      GO TO 670
  650 CALL FUNK
      NF#NF&1
      JVARY#I
C                                     SAVE OLD VALUE OF CHISQ FOR INTERPOLATION.
      CHIME#CHISQ
      IF%CHISQ-CHIOLD<810,660,670
  660 NFLAG#NFLAG&1
C                                     STEP X%I< THE OTHER WAY.
  670 X%I<#SAVE-DX%I<
      IF%X%I<-XMIN%I<<780,680,680
  680 IF%X%I<-XMAX%I<<690,690,780
  690 CALL FUNK
      NF#NF&1
      JVARY#I
      IF%CHISQ-CHIOLD<800,700,710
  700 NFLAG#NFLAG&1
  710 IF%NFLAG-3<720,760,780
C                                     PERFORM PARABOLIC INTERPOLATION.
C                                     CHECK FOR ZERO DENOMINATOR, ETC.
C
  720 IF%%CHISQ-CHIME<*%CHIME-2.*CHIOLD&CHISQ<<730,780,730
  730 TRIAL%I<#.5*DX%I<*%CHISQ-CHIME</%CHIME-2.*CHIOLD&CHISQ<
      VEC%I<#TRIAL%I</ADX
      X%I<#SAVE&TRIAL%I<
      CALL FUNK
      NF#NF&1
      IF%CHISQ-CHIOLD<740,750,750
```

```
  740 CHIOLD#CHISQ
      JOCK#1
      GO TO 790
  750 TRIAL%I<#0.0
      VEC%I<#0.0
      GO TO 780
  760 JFLAT%I<#1
  770 VEC%I<#-0.
  780 X%I<#SAVE
  790 NCIRC#NCIRC&1
      IF%NCIRC-NACTIV<910,1820,1820
C                                    RESET DX%I< FOR MORE EFFICIENT OPERATION.
  800 DX%I<#-DX%I<
C
C   A LOWER VALUE OF CHISQ HAS BEEN FOUND.  STEP, DOUBLE THE STEP SIZE,
C   AND REPEAT AS LONG AS CHISQ CONTINUES TO DECREASE.
C
  810 NCIRC#0
      DEL#DX%I<
      NSTP#0
  820 CHIME#CHIOLD
      CHIOLD#CHISQ
C     VEC%I<#VEC%I<&DEL/ABS %DX%I<<
      VEC%I<#VEC%I<&DEL/DABS%DX%I<<
      TRIAL%I<#TRIAL%I<&DEL
      NSTP#NSTP&1
      IF%NSTP-NSTPMX<830,890,890
  830 DEL#ACK*DEL
      SAVE#X%I<
      X%I<#SAVE&DEL
      IF%X%I<-XMIN%I<<900,840,840
  840 IF%X%I<-XMAX%I<<850,850,900
  850 CALL FUNK
      NF#NF&1
      IF%CHISQ-CHIOLD<820,860,860
C                                    PERFORM PARABOLIC INTERPOLATION.
  860 DENOM#ACK*CHIME-%ACK&1.<*CHIOLD&CHISQ
      IF%DENOM<870,900,870
  870 CINDER#%.5/ACK<*%ACK**2*CHIME-%ACK**2-1.<*CHIOLD-CHISQ</DENOM
      X%I<#SAVE&CINDER*DEL
      CALL FUNK
      NF#NF&1
      IF%CHISQ-CHIOLD<880,900,900
  880 CHIOLD#CHISQ
      TRIAL%I<#TRIAL%I<&CINDER*DEL
      VEC%I<#VEC%I<&CINDER*DEL/ADX
  890 JOCK#1
      GO TO 910
  900 X%I<#SAVE
C                                    DO NOT INCREASE THE STEP SIZE PREMATURELY.
  910 IF%NZIP<920,920,940
  920 IF%NCOMP-1<930,930,1720
  930 IF%NACK-1<1720,1720,940
C                                    AVEC#ABS%VEC%I<<
  940 AVEC#VEC%I<
      IF%AVEC<950,960,960
  950 AVEC#-AVEC
  960 IF%AVEC-FACUP<1030,970,970
  970 DX%I<#ACK*ADX
      VEC%I<#VEC%I</ACK
      OLDVEC%I<#OLDVEC%I</ACK
      IF%NOSC<1000,1000,980
  980 DO 990 J#1,NOSC
  990 ERR%I,J<#ERR%I,J</ACK
 1000 IF%NTRACE<1030,1030,1010
 1010 WRITE%          KW,1020<I,DX%I<
 1020 FORMAT%10H STEP SIZEI3,14H INCREASED TO E12.5<
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  IF POSSIBLE STEP ALONG A RESULTANT DIRECTION.
C
C                                    CHECK THE COLINEARITY OF VEC AND OLDVEC.
 1030 SUMO#0.0
      SUMV#0.0
      DO 1040 J#1,NV
      SUMO#SUMO&OLDVEC%J<**2
 1040 SUMV#SUMV&VEC%J<**2
      IF%SUMO*SUMV<1720,1720,1050
C1050 SUMO#SQRT %SUMO<
 1050 SUMO#DSQRT%SUMO<
C     SUMV#SQRT %SUMV<
      SUMV#DSQRT%SUMV<
      COSINE#0.0
```

```
      DO 1060 J#1,NV
 1060 COSINE#COSINE&%OLDVEC%J</SUMO<*%VEC%J</SUMV<
      IF%NZIP-NCOMP<1070,1100,1100
 1070 IF%NZIP-1<1720,1080,1090
 1080 IF%NACK-NACTIV<1720,1090,1090
 1090 IF%COSINE-COMPAR<1720,1100,1100
 1100 IF%VEC%I<<1110,1720,1110
 1110 NONZER#0
      DO 1130 J#1,NV
      IF%VEC%J<<1120,1130,1120
 1120 NONZER#NONZER&1
 1130 CONTINUE
      IF%NONZER-2<1720,1140,1140
C
C  SIMON SAYS, TAKE AS MANY GIANT STEPS AS POSSIBLE...
C
C                              NGIANT ...  NUMBER OF GIANT STEPS
 1140 NGIANT#0
C                              NTRY ...  NONZERO IF A GIGANTIC
C                                  %OSCILLATION< STEP IS BEING TRIED
      NTRY#0
C                              NRETRY ...  NUMBER OF OSCILLATION PERIODS
C                                  YET TO BE TESTED
      NRETRY#0
C                              KL ...  POINTER FOR OSCILLATION CHECK
      KL#1
C                              STORE OSCILLATION INFORMATION.
      NOSC#NOSC&1
      IF%NOSC-MOSQUE<1180,1180,1150
 1150 NOSC#MOSQUE
      IF%NOSC-1<1300,1180,1160
 1160 DO 1170 K#2,NOSC
      CHIOSC%K-1<#CHIOSC%K<
      DO 1170 J#1,NV
      XOSC%J,K-1<#XOSC%J,K<
 1170 ERR%J,K-1<#ERR%J,K<
C
 1180 CONTINUE
C
C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
C
C  TO DELETE THE OSCILLATION SEARCH SECTION, REMOVE ALL STATEMENTS
C  BETWEEN HERE AND THE NEXT COMMENT CARD OF X@S.
C
      DO 1190 J#1,NV
      XOSC%J,NOSC<#X%J<
 1190 ERR%J,NOSC<#VEC%J</SUMV
      CHIOSC%NOSC<#CHIOLD
      IF%NOSC-3<1300,1200,1200
C
C  SEARCH FOR A PREVIOUS SUCCESSFUL GIANT STEP IN A DIRECTION MORE
C  NEARLY PARALLEL TO THE DIRECTION OF THE PROPOSED STEP THAN WAS THE
C  IMMEDIATELY PREVIOUS ONE.  THIS MAY MEAN THAT THE DIRECTIONS OF THE
C  GIANT STEPS OSCILLATE PERIODICALLY.  TRY GIGANTIC %OSCILLATION<
C  STEPS OF DECREASING PERIOD, THEN ORDINARY GIANT STEPS.
C
 1200 COXCOM#0.0
      DO 1210 J#1,NV
 1210 COXCOM#COXCOM&ERR%J,NOSC<*ERR%J,NOSC-1<
      NAH#NOSC-2
 1220 NTRY#0
      DO 1240 K#KL,NAH
      NRETRY#NAH-K
      COSINE#0.0
      DO 1230 J#1,NV
 1230 COSINE#COSINE&ERR%J,NOSC<*ERR%J,K<
      IF%COSINE-COXCOM<1240,1240,1250
 1240 CONTINUE
      GO TO 1300
 1250 NTRY#1
      KL#K&1
      IF%NTRACE<1280,1280,1260
 1260 NT#NOSC-K
      WRITE%          KW,1270<NT,COXCOM,COSINE
 1270 FORMAT%/1X8H********5X26HGIGANTIC STEP WITH PERIOD I2,
     *    37H BEING ATTEMPTED.   COXCOM, COSINE # 2E13.4<
 1280 DO 1290 J#1,NV
      SALVO%J<#TRIAL%J<
 1290 TRIAL%J<#%X%J<-XOSC%J,K<</ACK
      CHIBAK#CHIOLD&%CHIOSC%K<-CHIOLD</ACK
      GO TO 1310
C
C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
C
```

```
C  PERFORM GIGANTIC OR GIANT STEPS.
C
 1300 CHIBAK#CHI%I<
 1310 DO 1360 J#1,NV
      XSAVE%J<#X%J<
      TRIAL%J<#ACK*TRIAL%J<
      IF%MASK%J<<1360,1320,1360
 1320 X%J<#X%J<&TRIAL%J<
C                                X%J<#AMAX1%AMIN1%X%J<,XMAX%J<<,XMIN%J<<
      IF%X%J<-XMAX%J<<1340,1340,1330
 1330 X%J<#XMAX%J<
 1340 IF%X%J<-XMIN%J<<1350,1360,1360
 1350 X%J<#XMIN%J<
 1360 CONTINUE
      JOCK#0
      JVARY#0
      CALL FUNK
      NF#NF&1
      IF%CHISQ-CHIOLD<1370,1440,1440
 1370 CHIBAK#CHIOLD
      CHIOLD#CHISQ
      NGIANT#NGIANT&1
      IF%NTRACE<1310,1310,1380
 1380 IF%NGIANT-1<1390,1390,1420
 1390 WRITE%               KW,1400<CHIBAK,NF,%VEC%J<,J#1,I<
 1400 FORMAT%//8H CHISQ #F16.8,8X4HNF #I7//5X16HNO. OF STEPS #  9E11.3/
     *     %21X9E11.3<<
      WRITE%               KW,1410<%XSAVE%J<,J#1,NV<
 1410 FORMAT%9H X%I<..../%1X9E13.5<<
 1420 WRITE%               KW,1430<CHISQ,NF,%X%J<,J#1,NV<
 1430 FORMAT%/8H CHISQ #E16.8,8X4HNF #I7/9H X%I<..../%1X9E13.5<<
      GO TO 1310
C                                DO NOT INTERPOLATE AFTER AN UNSUCCESSFUL
C                                       GIANT STEP.
 1440 IF%NGIANT<1530,1530,1450
C                                PERFORM PARABOLIC INTERPOLATION.
 1450 DENOM#ACK*CHIBAK-%ACK&1.<*CHIOLD&CHISQ
      IF%DENOM<1460,1520,1460
 1460 CINDER#%.5/ACK<*%ACK**2*CHIBAK-%ACK**2-1.<*CHIOLD-CHISQ</DENOM
      DO 1510 J#1,NV
      IF%MASK%J<<1510,1470,1510
 1470 X%J<#XSAVE%J<&CINDER*TRIAL%J<
C                                X%J<#AMAX1%AMIN1%X%J<,XMAX%J<<,XMIN%J<<
      IF%X%J<-XMAX%J<<1490,1490,1480
 1480 X%J<#XMAX%J<
 1490 IF%X%J<-XMIN%J<<1500,1510,1510
 1500 X%J<#XMIN%J<
 1510 CONTINUE
      JOCK#0
      JVARY#0
      CALL FUNK
      NF#NF&1
      IF%CHISQ-CHIOLD<1650,1520,1520
 1520 IF%NGIANT<1560,1530,1560
 1530 IF%NTRY<1540,1560,1540
 1540 DO 1550 J#1,NV
      TRIAL%J<#SALVO%J<
 1550 X%J<#XSAVE%J<
      GO TO 1580
 1560 DO 1570 J#1,NV
      TRIAL%J<#TRIAL%J</ACK
 1570 X%J<#XSAVE%J<
 1580 IF%NTRACE<1610,1610,1590
 1590 WRITE%               KW,1600<CHIOLD,NGIANT
 1600 FORMAT%/8H CHISQ #E15.8,7H  AFTERI3,13H GIANT STEPS.  <
      WRITE%               KW,1410<%X%J<,J#1,NV<
 1610 IF%NGIANT<1620,1620,1680
 1620 IF%NRETRY<1630,1630,1220
 1630 IF%NTRY<1640,1700,1640
C                                ALL GIGANTIC STEPS WERE UNSUCCESSFUL.
C                                TRY A GIANT STEP.
 1640 NTRY#0
      GO TO 1300
C
 1650 CHIOLD#CHISQ
      JOCK#1
      IF%NTRACE<1680,1680,1660
 1660 STEPS#NGIANT
      STEPS#STEPS&CINDER
      WRITE%               KW,1670<CHIOLD,STEPS
 1670 FORMAT%/8H CHISQ #E15.8,7H  AFTERF6.1,13H GIANT STEPS.   <
      WRITE%               KW,1410<%X%J<,J#1,NV<
 1680 IF%NTRY<1690,510,1690
 1690 CONTINUE
```

```
C                                A SUCCESSFUL GIGANTIC STEP HAS OCCURRED.    GRH05690
      GO TO 510                                                              GRH056A0
C                                AN UNSUCCESSFUL GIANT STEP HAS OCCURRED.    GRH056B0
C                                DELETE ITS OSCILLATION INFORMATION.         GRH056C0
C                                NOSC#MAX0%NOSC-1,0<                         GRH056D0
 1700 NOSC#NOSC-1                                                            GRH056E0
      IF%NOSC<1710,1720,1720                                                 GRH056F0
 1710 NOSC#0                                                                 GRH05700
 1720 CHI%I<#CHIOLD                                                          GRH05710
 1730 CONTINUE                                                               GRH05720
C                                DECOMMENT FOR SSW OPTION ....               GRH05730
C     CALL SSWTCH %NSSW,JUMP<                                                GRH05740
C     IF%JUMP-1<2090,2090,1740                                               GRH05750
C                                                                            GRH05760
 1740 IF%NF-NFMAX<1750,1750,2070                                             GRH05770
 1750 CONTINUE                                                               GRH05780
C                                END OF THE MAIN DO LOOP.                    GRH05790
C                                                                            GRH057A0
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *      GRH057B0
C                                                                            GRH057C0
C  ANOTHER CYCLE THROUGH THE VARIABLES HAS BEEN COMPLETED.                   GRH057D0
C  PRINT ANOTHER LINE OF TRACES.                                             GRH057E0
C                                                                            GRH057F0
      IF%NTRACE<1770,1770,1760                                               GRH05800
 1760 WRITE%           KW,1400<CHIOLD,NF,%VEC%J<,J#1,NV<                     GRH05810
 1770 IF%NZIP<1810,1780,1810                                                 GRH05820
 1780 IF%NTRACE<1810,1810,1790                                               GRH05830
 1790 WRITE%           KW,1410<%X%J<,J#1,NV<                                 GRH05840
      WRITE%           KW,1800<                                              GRH05850
 1800 FORMAT%1H <                                                            GRH05860
 1810 NZIP#NZIP&1                                                            GRH05870
      GO TO 520                                                              GRH05880
C                                                                            GRH05890
C  A MINIMUM HAS BEEN FOUND.  PRINT THE REMAINING TRACES.                    GRH058A0
C                                                                            GRH058B0
 1820 IF%NTRACE<1840,1840,1830                                               GRH058C0
 1830 WRITE%           KW,1400<CHIOLD,NF,%VEC%J<,J#1,I<                      GRH058D0
      WRITE%           KW,1410<%X%J<,J#1,NV<                                 GRH058E0
C                                                                            GRH058F0
C  DECREASE THE SIZE OF THE STEPS FOR ALL VARIABLES.                         GRH05900
C                                                                            GRH05910
 1840 CONTINUE                                                               GRH05920
C                                DECOMMENT FOR SSW OPTION ....               GRH05930
C     CALL SSWTCH %NSSW,JUMP<                                                GRH05940
C     IF%JUMP-1<2090,2090,1850                                               GRH05950
C                                                                            GRH05960
 1850 IF%NF-NFMAX<1860,1860,2070                                             GRH05970
C                                CHECK WHETHER ALL DX%J< .LE. DELMIN%J<.     GRH05980
 1860 NGATE#1                                                                GRH05990
      DO 1910 J#1,NV                                                         GRH059A0
      IF%MASK%J<<1910,1870,1910                                              GRH059B0
C                                ADX#ABS%DX%J<<                              GRH059C0
 1870 ADX#DX%J<                                                              GRH059D0
      IF%ADX<1880,1890,1890                                                  GRH059E0
 1880 ADX#-ADX                                                               GRH059F0
 1890 IF%ADX-DELMIN%J<<1910,1910,1900                                        GRH05A00
 1900 NGATE#0                                                                GRH05A10
 1910 DX%J<#DX%J</RATIO                                                      GRH05A20
      IF%NGATE<1920,1920,1960                                                GRH05A30
C                                CHECK THE JFLAT%J<.                         GRH05A40
 1920 JFLMIN#5                                                               GRH05A50
      DO 1950 J#1,NV                                                         GRH05A60
      IF%MASK%J<<1950,1930,1950                                              GRH05A70
 1930 IF%JFLAT%J<-JFLMIN<1940,1950,1950                                      GRH05A80
 1940 JFLMIN#JFLAT%J<                                                        GRH05A90
 1950 CONTINUE                                                               GRH05AA0
      IF%JFLMIN-1<2040,1990,1960                                             GRH05AB0
 1960 IF%NTRACE<2130,1970,1970                                               GRH05AC0
 1970 WRITE%           KW,1980<                                              GRH05AD0
 1980 FORMAT%///65H TERMINATED WHEN THE STEP SIZES BECAME AS SMALL AS THGRH05AE0
     *E DELMIN%J<.    <                                                      GRH05AF0
      GO TO 2130                                                             GRH05B00
 1990 IF%NFLAT<2040,2040,2000                                                GRH05B10
 2000 IF%NTRACE<2130,2010,2010                                               GRH05B20
 2010 WRITE%           KW,2020<                                              GRH05B30
 2020 FORMAT%///72H TERMINATED WHEN THE FUNCTION VALUES AT ALL TRIAL POIGRH05B40
     *NTS WERE IDENTICAL.   <                                                GRH05B50
      WRITE%           KW,2030<%DX%J<,J#1,NV<                                GRH05B60
 2030 FORMAT%///29H CURRENT STEP SIZE VALUES....//%1X9E13.5<<                GRH05B70
      GO TO 2130                                                             GRH05B80
 2040 IF%NTRACE<510,510,2050                                                 GRH05B90
C                                PRINT THE DX%J< AND SEARCH SOME MORE.       GRH05BA0
C                                                                            GRH05BB0
 2050 WRITE%           KW,2060<%DX%J<,J#1,NV<                                GRH05BC0
```

```
 2060 FORMAT%//60%1X1H*<//26H STEP SIZES REDUCED TO....//%1X9E13.5<<       GRH05BD0
      GO TO 510                                                             GRH05BE0
C                                                                           GRH05BF0
 2070 WRITE%              KW,2080<NFMAX                                     GRH05C00
 2080 FORMAT%///46H ABNORMAL TERMINATION....    MORE THAN NFMAX # I7,       GRH05C10
     *   31H CALLS TO THE CHISQ SUBROUTINE.<                                GRH05C20
      GO TO 2110                                                            GRH05C30
C                               DECOMMENT FOR SSW OPTION ....               GRH05C40
 2090 WRITE%              KW,2100<                                          GRH05C50
 2100 FORMAT%///42H SUBROUTINE STEPIT TERMINATED BY OPERATOR.<              GRH05C60
C                                                                           GRH05C70
 2110 WRITE%              KW,2030<%DX%J<,J#1,NV<                            GRH05C80
C                                                                           GRH05C90
C                               SET SWITCH FOR TERMINATION.                 GRH05CA0
 2120 KWIT#1                                                                GRH05CB0
C                                                                           GRH05CC0
C                               CALL FUNK WITH THE BEST SET OF X%J<.        GRH05CD0
 2130 JVARY#0                                                               GRH05CE0
      CALL FUNK                                                             GRH05CF0
      IF%CHISQ-CHISAV<2150,2150,2140                                        GRH05D00
 2140 WRITE %KW,410< CHISAV,CHISQ,NF                                        GRH05D10
 2150 IF%NTRACE<2200,2160,2160                                              GRH05D20
 2160 WRITE%              KW,2170<NF                                        GRH05D30
 2170 FORMAT%// 1X,I5,23H FUNCTION COMPUTATIONS         <                   GRH05D40
      WRITE%              KW,2180<%X%J<,J#1,NV<                             GRH05D50
 2180 FORMAT%///10X24HFINAL VALUES OF X%I<....//%1X5E21.13<<                GRH05D60
      WRITE%              KW,2190<CHISQ                                     GRH05D70
 2190 FORMAT%//24H FINAL VALUE OF CHISQ # E21.13//<                         GRH05D80
 2200 IF%KWIT<2250,2210,2250                                                GRH05D90
C                               MATD#IABS%MATRIX-100<                       GRH05DA0
 2210 MATD#MATRIX-100                                                       GRH05DB0
      IF%MATD<2220,2230,2230                                                GRH05DC0
 2220 MATD#-MATD                                                            GRH05DD0
 2230 IF%MATD-50<2240,2240,2250                                             GRH05DE0
C                                                                           GRH05DF0
C                               SKIP ERROR CALCULATION IF ANY MASK%J<.NE.0.GRH05E00
 2240 IF%NACTIV-NV<2250,2260,2250                                           GRH05E10
 2250 RETURN                                                                GRH05E20
C                                                                           GRH05E30
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     GRH05E40
C                                                                           GRH05E50
 2260 CONTINUE                                                              GRH05E60
C                                                                           GRH05E70
C X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X     GRH05E80
C                                                                           GRH05E90
C   TO DELETE THE ERROR COMPUTATION SECTION, REMOVE ALL STATEMENTS          GRH05EA0
C   BETWEEN HERE AND THE END STATEMENT, AND INSERT A RETURN STATEMENT.      GRH05EB0
C                                                                           GRH05EC0
C   COMPUTE THE STANDARD ERRORS AND THE CORRELATIONS.                       GRH05ED0
C   FIRST COMPUTE THE SECOND PARTIAL DERIVATIVES OF CHISQ WITH RESPECT      GRH05EE0
C   TO THE X%J<.                                                            GRH05EF0
C                                                                           GRH05F00
      FAC#RATIO**%MATRIX-100<                                               GRH05F10
C                               COMPUTE THE DIAGONAL PARTIALS FIRST.        GRH05F20
      DO 2300 I#1,NV                                                        GRH05F30
C                               DX%I<#ABS%FAC*DX%I<<                        GRH05F40
      DX%I<#FAC*DX%I<                                                       GRH05F50
      IF%DX%I<<2270,2280,2280                                               GRH05F60
 2270 DX%I<#-DX%I<                                                          GRH05F70
 2280 XSAVE%I<#X%I<                                                         GRH05F80
      JVARY#0                                                               GRH05F90
      DO 2290 J#1,2                                                         GRH05FA0
      X%I<#XSAVE%I<&DX%I<                                                   GRH05FB0
      CALL FUNK                                                             GRH05FC0
      NF#NF&1                                                               GRH05FD0
      JVARY#I                                                               GRH05FE0
      SECOND%1,J<#CHISQ                                                     GRH05FF0
 2290 DX%I<#-DX%I<                                                          GRH06000
      X%I<#XSAVE%I<                                                         GRH06010
 2300 ERR%I,I<#%SECOND%1,1<-2.0*CHIOLD&SECOND%1,2<</DX%I<**2                GRH06020
C                                                                           GRH06030
C                               COMPUTE THE OFF-DIAGONAL PARTIALS.  USE A   GRH06040
C                                     REDUNDANT FOUR-POINT RULE FOR GREATER GRH06050
C                                     RELIABILITY.                          GRH06060
      IF%NV-2<2350,2310,2310                                                GRH06070
 2310 DO 2340 I#2,NV                                                        GRH06080
      IM#I-1                                                                GRH06090
      DO 2340 J#1,IM                                                        GRH060A0
      DO 2330 K#1,2                                                         GRH060B0
      X%I<#XSAVE%I<&DX%I<                                                   GRH060C0
      JVARY#0                                                               GRH060D0
      DO 2320 L#1,2                                                         GRH060E0
      X%J<#XSAVE%J<&DX%J<                                                   GRH060F0
      CALL FUNK                                                             GRH06100
```

```
      NF#NF&1
      JVARY#J
      SECOND%K,L<#CHISQ
      X%J<#XSAVE%J<
 2320 DX%J<#-DX%J<
      X%I<#XSAVE%I<
C                                DECOMMENT FOR SSW OPTION ....
C     CALL SSWTCH %NSSW,JUMP<
C     IF%JUMP-1<2090,2090,2330
C
 2330 DX%I<#-DX%I<
      ERR%I,J<#0.25*%SECOND%1,1<-SECOND%1,2<-SECOND%2,1<&SECOND%2,2<<
     *    /%DX%I<*DX%J<<
 2340 ERR%J,I<#ERR%I,J<
C                                END OF DERIVATIVE COMPUTATION.
C
 2350 IF%NTRACE<2410,2360,2360
 2360 WRITE%                 KW,2370<
 2370 FORMAT%41H1SIZES OF INCREMENTS TO BE USED BELOW....<
      WRITE%                 KW,2380<%DX%J<,J#1,NV<
 2380 FORMAT%/%1X9E13.5<<
      WRITE%                 KW,2390<
 2390 FORMAT%/////45H MATRIX OF THE SECOND PARTIAL DERIVATIVES.... /<
      DO 2400 I#1,NV
 2400 WRITE%                 KW,2380<%ERR%I,J<,J#1,I<
 2410 DO 2420 I#1,NV
      DO 2420 J#1,I
      IF%ERR%I,J<<2420,2430,2420
 2420 CONTINUE
      GO TO 2450
 2430 WRITE%                 KW,2440<
 2440 FORMAT%////46H THE ABOVE MATRIX CONTAINS ONE OR MORE ZEROES. /
     *     75H A LARGER VALUE OF -MATRIX- SHOULD BE TRIED, TO SEE IF THEY
     * ARE LEGITIMATE.    <
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  INVERT THE MATRIX OF SECOND DERIVATIVES USING THE ALGORITHM SYMINV2
C  %ALGORITHM 150, H. RUTISHAUSER, COMMUNICATIONS OF THE A.C.M. 6 %1963<
C  P. 67<. RE-USE SOME STORAGE.
C
 2450 SGNDET#1.
      DETLOG#0.
      DO 2460 J#1,NV
 2460 SALVO%J<#1.0
      DO 2660 I#1,NV
C                                PERFORM PIVOT SEARCH.
      BIGAJJ#0.0
      DO 2510 J#1,NV
      IF%SALVO%J<<2470,2510,2470
C                                ABER#ABS%ERR%J,J<<
 2470 ABER#ERR%J,J<
      IF%ABER<2480,2490,2490
 2480 ABER#-ABER
 2490 IF%ABER-BIGAJJ<2510,2510,2500
 2500 BIGAJJ#ABER
      K#J
 2510 CONTINUE
      IF%BIGAJJ<2540,2520,2540
 2520 WRITE%                 KW,2530<
 2530 FORMAT%////67H ERROR MATRIX IS SINGULAR.  -MATRIX- SHOULD PROBABLY
     * BE INCREASED.            /////<
      GO TO 2120
 2540 SALVO%K<#0.0
      IF%ERR%K,K<<2550,2550,2560
 2550 SGNDET#-SGNDET
C2560 DETLOG#DETLOG&ALOG%BIGAJJ</2.303
 2560 DETLOG#DETLOG&DLOG%BIGAJJ</2.303
C
C                                PREPARE FOR THE NEXT ELIMINATION STEP.
      TRIAL%K<#1.0/ERR%K,K<
      ERR%K,K<#0.0
      XSAVE%K<#1.0
      M#K-1
      IF%M<2600,2600,2570
 2570 DO 2590 J#1,M
      XSAVE%J<#ERR%K,J<
      TRIAL%J<#ERR%K,J<*TRIAL%K<
      IF%SALVO%J<<2520,2590,2580
 2580 TRIAL%J<#-TRIAL%J<
 2590 ERR%K,J<#0.0
 2600 M#K&1
      IF%M-NV<2610,2610,2650
 2610 DO 2640 J#M,NV
```

```
      XSAVE%J<#ERR%J,K<
      IF%SALVO%J<<2520,2620,2630
 2620 XSAVE%J<#-XSAVE%J<
 2630 TRIAL%J<#-ERR%J,K<*TRIAL%K<
 2640 ERR%J,K<#0.0
C                                    PERFORM THE NEXT ELIMINATION STEP.
 2650 DO 2660 J#1,NV
      DO 2660 K#J,NV
 2660 ERR%K,J<#ERR%K,J<&XSAVE%J<*TRIAL%K<
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  PRINT THE ERRORS AND CORRELATIONS, AND TERMINATE.
C
      IF%SGNDET<2670,2690,2690
 2670 WRITE%              KW,2680<
 2680 FORMAT%////75H ERROR MATRIX IS NEGATIVE DEFINITE.  -MATRIX- SHOULD
     * PROBABLY BE DECREASED.    <
 2690 IF%NTRACE<2720,2700,2700
 2700 WRITE %KW,2710< DETLOG,SGNDET
 2710 FORMAT%////39H LOG10F%DETERMINANT OF ABOVE MATRIX< # E12.5,10X
     *    22HSIGN OF DETERMINANT # E8.1<
C                                    THE ERROR MATRIX IS TWICE THE INVERSE OF
C                                        THE MATRIX OF SECOND DERIVATIVES.
 2720 DO 2790 I#1,NV
      DO 2730 J#1,I
      ERR%I,J<#ERR%I,J<*2.0
 2730 ERR%J,I<#ERR%I,J<
C                                    XSAVE%I<#SIGN%SQRT%ABS%ERR%I,I<<<,ERR%I,I<<
      ABER#ERR%I,I<
      IF%ABER<2740,2790,2750
 2740 ABER#-ABER
C2750 ABER#SQRT %ABER<
 2750 ABER#DSQRT%ABER<
      IF%ERR%I,I<<2760,2770,2790
 2760 ABER#-ABER
 2770 WRITE%              KW,2780<ERR%I,I<
 2780 FORMAT% ///50H NEGATIVE OR ZERO MEAN SQUARE ERROR ENCOUNTERED...
     *    3XE15.8/39H -MATRIX- SHOULD PROBABLY BE DECREASED. ///<
 2790 XSAVE%I<#ABER
      IF%NTRACE<2130,2800,2800
 2800 WRITE%              KW,2810<
 2810 FORMAT%/////20H STANDARD ERRORS.... <
      WRITE%              KW,2380<%XSAVE%J<,J#1,NV<
      IF%NV-1<2120,2120,2820
 2820 WRITE%              KW,2830<
 2830 FORMAT%/////45H LOWER TRIANGLE OF THE CORRELATION MATRIX.... /<
      DO 2880 I#2,NV
      IM#I-1
      DO 2870 J#1,IM
      DENOM#XSAVE%I<*XSAVE%J<
      IF%DENOM<2850,2840,2860
 2840 TRIAL%J<#0.
      GO TO 2870
 2850 DENOM#-DENOM
 2860 TRIAL%J<#ERR%I,J</DENOM
 2870 CONTINUE
 2880 WRITE%              KW,2380<%TRIAL%J<,J#1,IM<
      GO TO 2120
      END
      BLOCK DATA
C
C  BLOCK DATA SUBPROGRAM FOR STEPIT, SIMPLEX, AND STP.
C  J. P. CHANDLER, F.S.U. PHYSICS DEPT.
C
      COMMON /FRODO/ NFMAX,NFLAT,JVARY,NEXTRA
      COMMON/ HISTO / ICURVE,CRVMAX,ABE%102<,BIN%102<
      DATA NFMAX/1000000/, NFLAT/1/, NEXTRA/0/
      DATA  ICURVE/0/, CRVMAX/0./
      END
/*
//
```

E.5 SUBROUTINE USED FOR GENERATING A GREAT CIRCLE ARC

```
C       MAIN
C
C       THIS PROGRAM GENERATES A TEST ARC OF VARIABLE LENGTH IN
C       CARTESIAN COORDINATES, PROJECTS AND TRANSFORMS TO DATA BASE
C       COORDINATES AND COMPARES DIRECT AND INTERPOLATED VALUES FOR
C       INTERMEDIATE POINTS ON THE ARC.
C
C       PROGRAMMER - E. MICHAEL O'NEILL
C                    COMPUTER SCIENCES CORPORATION
C                    8728 COLESVILLE RD. SILVER SPRING, MD.
C
      DIMENSION XYZO(3), XYZ(3), BXYZ(3)
      DATA XYZO,PI/1.0,0.0,0.0,3.14159/
      TARC=0.02
      DARC=PI/16.0
      ANG=0.0
      ARC=0.0
      BANG=PI/2.0
      DO 10 I=1,5
      CALL CIRARC(XYZO,ANG,ARC,BXYZ)
      ARK=0.0
      DO 5 J=1,5
      WRITE(6,4) I,J
    4 FORMAT(' I,J ',2I3/)
      CALL CIRARC(BXYZ,BANG,ARK,XYZ)
      CALL TSTARC(XYZ,TARC)
    5 ARK=ARK+DARC
   10 ARC=ARC+DARC
      STOP
      END
      SUBROUTINE TSTARC(XYZO,ARCO)
      DIMENSION XYZO(3),XYZ1(3),XYZ(3)
      DATA PI/3.14159/
      DANG=PI/16.0
      DARC=ARCO/4.0
      ANG=PI
      DO 20 IANG=1,3
      CALL CIRARC(XYZO,ANG,ARCO,XYZ1)
      CALL QTRAN(XYZO,PO,EO,IFO)
      CALL QTRAN(XYZ1,P1,E1,IF1)
      DP=(P1-PO)/4.0
      DE=(E1-EO)/4.0
      P=PO
      E=EO
      ARC=0.0
      DO 10 IARC=1,5
      CALL CIRARC(XYZO,ANG,ARC,XYZ)
      CALL QTRAN(XYZ,PC,EC,IFC)
      WRITE(6,5) PC,P,EC,E,IFC,XYZ
    5 FORMAT(1X,4F12.3,I12,3F12.3)
      P=P+DP
      E=E+DE
   10 ARC=ARC+DARC
      WRITE(6,5)
   20 ANG=ANG+DANG
      WRITE(6,5)
      RETURN
      END
      SUBROUTINE CIRARC(XYZO,ANG,ARC,XYZ)
      DIMENSION XYZO(3), XYZ(3)
      RAD=SQRT(XYZO(1)**2+XYZO(2)**2+XYZO(3)**2)
      RHO=SQRT(XYZO(1)**2+XYZO(2)**2)
      C1=COS(ANG)
      S1=SIN(ANG)
      C2=COS(ARC)
      S2=SIN(ARC)
      XYZ(1)=(XYZO(1)*XYZO(3)*C1-RAD*XYZO(2)*S1)*S2/RHO + XYZO(1)*C2
      XYZ(2)=(XYZO(2)*XYZO(3)*C1+RAD*XYZO(1)*S1)*S2/RHO + XYZO(2)*C2
      XYZ(3)= -RHO*C1*S2 + XYZO(3)*C2
      RETURN
      END
```

E.6 SUBROUTINES USED FOR MAPPING A POINT IN CARTESIAN SPACE INTO DATA BASE COORDINATES

```
      SUBROUTINE QTRAN(COORD,X,Y,IFACE)
C
C       THIS SUBROUTINE TAKES AS INPUT THE CARTESIAN COORDINATE ARRAY
C       (COORD) AND RETURNS DATA BASE COORDINATES (X AND Y) AND THE
C       FACE NUMBER (IFACE) OF THE X,Y SYSTEM
C
C       PROGRAMMER - E. MICHAEL O'NEILL
C                    COMPUTER SCIENCES CORPORATION
C                    8728 COLESVILLE RD. SILVER SPRING, MD.
C
      COMMON /TRACOM/ RS,RO,RSO,R4,GAMMA,OMGAM,DELTA,OMEGA,
     1                C00,C10,C01,C11,C20,C02,D0,D1,D2
      DIMENSION COORD(3), C(3), ROT(3,6), IROT(3,6)
      DATA ROT/1.0,1.0,1.0, -1.0,1.0,-1.0, -1.0,1.0,1.0,
     1           1.0,1.0,-1.0, -1.0,-1.0,1.0, -1.0,1.0,-1.0/
      DATA IROT /2,3,1,2,3,1,1,3,2,1,3,2,1,2,3,1,2,3/
      DATA OFFSET/2048.0/
C       DETERMINE FACE NUMBER BY FINDING LARGEST (ABS) COMPONENT
      CMAX=ABS(COORD(1))
      J=1
      DO 10 I=2,3
      IF(CMAX.LT.(ABS(COORD(I)))) J=I
      CMAX=ABS(COORD(J))
   10 CONTINUE
      IFACE=J*2
      IF(COORD(J).GE.0.0) IFACE=IFACE-1
C       INTERCHANGE CARTESIAN COMPONENTS TO GET TO FACE SYSTEM
      DO 20 I=1,3
   20 C(I)=COORD(IROT(I,IFACE))*ROT(I,IFACE)
C       PROJECT TO PSI, ETA SYSTEM
      PRJ=RO/C(3)
      P=C(1)*PRJ
      E=C(2)*PRJ
C       TRANSFORM TO X,Y
      X=FMAP(P,E)/RO*OFFSET+OFFSET
      Y=FMAP(E,P)/RO*OFFSET+OFFSET
      RETURN
      END
```

```
      FUNCTION FMAP(A,B)
C
C      THIS FUNCTION PERFORMS THE INVERSE TRANSFORMATION (F STAR)
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      COMMON /TRACOM/ RS,RO,RSQ,R4,GAMMA,OMGAM,DELTA,OMEGA,
     1               COO,C10,C01,C11,C20,C02,D0,D1,D2
      ASQ=A*A
      A3=A*ASQ
      A4=ASQ*ASQ
      BSQ=B*B
      B4=BSQ*BSQ
      RMA=RSQ-ASQ
      FMAP=A*GAMMA+A3*(OMGAM/RSQ)+A*BSQ*RMA*(DELTA+(RSQ-BSQ)*
     1     (C00+C10*ASQ+C01*BSQ+C11*ASQ*BSQ+C20*A4+C02*B4))+
     2     A3*RMA*(OMEGA+RMA*(D0+D1*ASQ+D2*A4))
      RETURN
      END
      BLOCK DATA
C      DATA INITIALIZATION FOR FMAP
C
      COMMON /TRACOM/ RS,RO,RSQ,R4,GAMMA,OMGAM,DELTA,OMEGA,
     1               C00,C10,C01,C11,C20,C02,D0,D1,D2
      DATA RS,RO,RSQ,R4/1.0,0.577350,0.333333,0.111111/
      DATA GAMMA,OMGAM,DELTA,OMEGA/1.381976,-.381976,-.834527,0.056318/
      DATA C00,C10,C01,C11,C20,C02/4.800983,23.47334,-40.99963,
     1                           325.4193,-318.1992,-9.85862/
      DATA D0,D1,D2/-12.80531,59.46511,-256.5819/
      END
```

E.7 SUBROUTINES USED FOR SIMULATING THE FAST-FILLING METHOD

```
      SUBROUTINE SCAN1(INBASE,NSL,NBL)
C
C      THIS SUBROUTINE SIMULATES THE STORAGE OF DATA IN THE DATA BASE
C      ON A SINGLE INTERPOLATION BLOCK BASIS.
C      INTERPOLATION ALONG AND BETWEEN SCAN LINES TAKES PLACE USING
C      VECTOR COMPONENTS INPUTTED (IPZ,IEZ,IDP,IDE,JDP,JDE).
C      THE SERIAL ADDRESS IS CALCULATED USING A TABLE LOOK UP AND
C      THE RECORD NUMBER IS DETERMINED BY A SHIFT OF THE SERIAL ADDRESS.
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL FAILW
      LOGICAL*1 INDATA,IODATA
      INTEGER SHFTL,SHFTR
      COMMON/DATCOM/ INDATA(81920), IODATA(24576)
      COMMON /INDCOM/ INDX(4096), INDY(4096), ISHFT, NMIL
      COMMON /SCNCOM/ IPZ, IEZ, IDP, IDE, JDP, JDE, JSIZE
      DATA LREC/-1/
      KLOC=INBASE
      DO 20 IBL=1,NBL
      JLOC=KLOC
      IP=IPZ
      IE=IEZ
      DO 10 ISL=1,NSL
      I=IP/ISHFT
      J=JP/ISHFT
      KEY=INDX(I)+INDY(J)
      IREC=SHFTR(KEY,6)
      IF(IREC.EQ.LREC) GO TO 5
      LREC=IREC
      CALL IOMNGR(IREC,IOBASE,FAILW)
    5 IF(FAILW) GO TO 9
      ILOC=KEY-SHFTL(IREC,6)
      IDUM=IOBASE+ILOC
C     IODATA(IOBASE+ILOC)=INDATA(JLOC)
    9 IP=IP+IDP
   10 IE=IE+IDE
      KLOC=KLOC+JSIZE
      IPZ=IPZ+JDP
   20 IEZ=IEZ+JDE
      RETURN
      END
      SUBROUTINE IOMNGR(IREC,IOBASE,FAILW)
      LOGICAL FAILW
      IOBASE=0
      FAILW=.FALSE.
      RETURN
      END
//  EXEC LOADER,REGION=200K,PARM='SIZE=200000'
//
```

E.8 SUBROUTINES USED FOR TESTING I/O HANDLING METHOD

```
      SUBROUTINE IOMNGR(IKEY,IOBASE,FAILW)
C
C      THIS SUBROUTINE CONTROLS DATA BASE INPUT/OUTPUT
C
CC      THIS SUBROUTINE TAKES AS INPUT THE SERIAL ADDRESS OF A
C      RECORD REQUIRED FOR PROCESSING, COMPARES THE RECORD NUMBER
C      WITH A LIST OF RECORDS AVAILABLE IN CORE BUFFER AND FETCHES
C      NEW RECORDS WHEN NEEDED.
C      THE CORE BUFFER IS CONTROLED BY A PUSH DOWN STACK,
C      WHEN A NEW RECORD IS REQUIRED THE OLDEST RECORD IN THE BUFFER
C      IS RETURNED TO THE DATA BASE AND REPLACED BY THE REQUESTED
C      RECORD. THE BASE ADDRESS OF THE REQUESTED RECORD IS RETURNED
C      TO THE CALLING PROGRAM IN IOBASE.
C      IOMNGR ALSO RETURNS THE WINDOW FLAG TO INDICATE PRESENCE
C      OR ABSENCE OF THE REQUESTED RECORD IN THE DATA BASE.
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL FAILW, FAIL(3,6)
      DIMENSION INDEX(3,6), LNDEX(3)
      EQUIVALENCE (FAIL(1,1),INDEX(1,1))
      DATA NSTAK/0/, MSTAK/6/
      DATA INDEX /-1,6,0,-1,5,0,-1,4,0,-1,3,0,-1,2,0,-1,1,0/
C
C      ON FIRST ENTRY IMMEDIATELY GET REQUESTED RECORD
C
      IF(NSTAK.EQ.0) GO TO 25
C
C      TEST REQUEST RECORD NUMBER AGAINST RECORDS IN CORE
C
      DO 20 I=1,NSTAK
      JKEY=INDEX(2,I)
C
C      BRANCH IF A MATCH IS FOUND
C
      IF(IKEY.EQ.INDEX(1,I)) GO TO 30
   20 CONTINUE
C
C      BRANCH IF STACK IS NOT YET FULL
C
      IF(NSTAK.LT.MSTAK) GO TO 25
C
C      ENTER REQUESTED RECORD INTO STACK
C
      CALL PUSHD(INDEX,LNDEX,MSTAK,IKEY)
      JKEY=LNDEX(2)
C
C      RETURN OLDEST RECORD TO DATA BASE
C
      CALL PUTREC(LNDEX(1),LNDEX(2),FAILW)
C
C      READ NEW RECORD INTO SPACE VACATED
C
      CALL GETREC(INDEX(1,1),INDEX(2,1),FAILW)
      FAIL(3,JKEY)=FAILW
      GO TO 30
C
C      ENTER AND FETCH ONLY UNTIL STACK IS FULL
C
   25 NSTAK=NSTAK+1
      JKEY=NSTAK
      CALL PUSHD(INDEX,LNDEX,MSTAK,IKEY)
      CALL GETREC(IKEY,JKEY,FAILW)
      FAIL(3,JKEY)=FAILW
C
C      SET BASE ADDRESS AND WINDOW FAILURE FLAG BEFORE RETURNING
C
   30 IOBASE=(JKEY-1)*1024
      FAILW=FAIL(3,JKEY)
      RETURN
      END
```

```
      SUBROUTINE PUSHD(ISTACK,L,MAX,IN)
C
C      PUSH DOWN STACK ROUTINE
C
C      CONTROLS TRIPLE STACK CONTAINING CURRENTLY AVAILABLE RECORD
C      NUMBERS AND THE ASSOCIATED BUFFER POINTER AND WINDOW FLAG
C        ISTACK IS AN ARRAY CONTAINING THE STACK
C        L IS THE VECTOR ELEMENT OF ISTACK BEING REPLACED BY IN
C        MAX IS THE DEPTH OF THE STACK
C        IN IS THE NEW SCALAR ADDED TO THE STACK
C       IN IS AUTOMATICALLY ASSOCIATED WITH THE POINTER FREED BY L
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      DIMENSION ISTACK(3,1), L(3)
      L(1)=ISTACK(1,MAX)
      L(2)=ISTACK(2,MAX)
      L(3)=ISTACK(3,MAX)
      K=MAX-1
      J=MAX
      DO 10 I=1,K
      ISTACK(1,J)=ISTACK(1,J-1)
      ISTACK(2,J)=ISTACK(2,J-1)
      ISTACK(3,J)=ISTACK(3,J-1)
   10 J=J-1
      ISTACK(1,1)=IN
      ISTACK(2,1)=L(2)
      RETURN
      END
      SUBROUTINE GETREC(NREC,IPOS,FAILW)
C
C      THIS SUBROUTINE HANDLES THE ACTUAL READING AND WRITING OF
C      DATA BASE RECORDS. NREC, THE RECORD NUMBER AND IPOS, THE
C      BUFFER BASE ADDRESS ARE INPUT. SUBROUTINE WINDOW IS CALLED TO
C      ASCERTAIN WHETHER OR NOT THE RECORD IS IN THE DATA BASE AND ITS
C      ABSOLUTE ADDRESS, MREC. FAILW, THE WINDOW FLAG, IS RETURNED
C      TO THE CALLING PROGRAM.
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL FAILW
      COMMON IDATA(6144)
      DATA NDBASE/11/
C
C      CALL WINDOW TO OBTAIN ABSOLUTE RECORD NUMBER AND WINDOW FLAG
C
      CALL WINDOW(NREC,MREC,FAILW)
C      RETURN IF RECORD DOES NOT EXIST
      IF(FAILW) RETURN
C      CALCULATE BASE ADDRESS AND READ RECORD
      I=(IPOS-1)*1024+1
      L=IPOS+1023
      READ(NDBASE'MREC) (IDATA(J),J=I,L)
      RETURN
      ENTRY PUTREC(NREC,IPOS,FAILW)
C
C      CALL WINDOW TO OBTAIN ABSOLUTE RECORD NUMBER AND WINDOW FLAG
C
      CALL WINDOW(NREC,MREC,FAILW)
C      RETURN IF RECORD DOES NOT EXIST
      IF(FAILW) RETURN
C      CALCULATE BASE ADDRESS AND WRITE OUT RECORD
      I=(IPOS-1)*1024+1
      L=IPOS+1023
      WRITE(NDBASE'MREC) (IDATA(J),J=I,L)
      RETURN
      END
```

```
      SUBROUTINE WINDOW(NREC,MREC,FAILW)
C
C      THIS SUBROUTINE WINDOWS THE DATA BASE AND ASSIGNS INDIRECT
C      RECORD NUMBERS TO THE ACTUAL PHYSICAL RECORDS
C
C      A CALL TO SETWIN  INITIALIZES THE RECORD NUMBER (IREC) AND
C      FAILURE FLAG (WIND) TABLES.
C      A CALL TO WINDOW WITH NREC AS INPUT RESULTS IN  THE RETURN
C      AN ABSOLUTE RECORD NUMBER (MREC) AND A WINDOW FAILURE FLAG
C      (FAILW).
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL FAILW
      LOGICAL*1 WIND(1536)
      DIMENSION IREC(1536)
      DATA WIND/1536*.TRUE./
      DATA IREC/1536*0/
      FAILW=.FALSE.
C      SET FLAG TO INDICATE RECORD IS PRESENT
C      CALCULATE WHICH RECORD BLOCK REQUESTED RECORD BELONGS TO
      JREC=(NREC-1)/16+1
      KREC=NREC-JREC*16
C      LOOK UP ADDRESS IN ABSOLUTE ADDRESS TABLE
      MREC=IREC(JREC)*16+KREC
C      SET FAILURE FLAG IF RECORD DOES NOT EXIST
      IF(WIND(JREC)) FAILW=.TRUE.
      RETURN
      ENTRY SETWIN
C      SET UP DATA BASE AREAS TO BE STORED, WIND=.FALSE. MEANS
C      THAT THE RECORD BLOCK EXISTS.
      WIND(1)=.FALSE.
      WIND(2)=.FALSE.
      WIND(5)=.FALSE.
      WIND(6)=.FALSE.
C      DERIVE ABSOLUTE ADDRESS TABLE FROM TABLE OF NON EXISTANT RECORDS.
      J=0
      DO 10 I=1,1536
      IF(.NOT.WIND(I)) J=J+1
      IF(.NOT.WIND(I)) IREC(I)=J
   10 CONTINUE
      RETURN
      END
```

E.9 SUBROUTINES USED FOR RETRIEVING DATA FROM THE DATA BASE AND FOR GRAPHIC DISPLAY

```
      SUBROUTINE RASTER(ISER,IBLOK)
C
C      THIS SUBROUTINE UNPACKS FROM SERIAL TO RASTER FOR THE SIXTH
C      LEVEL OF DIVISION ( 4096 - 64 X 64 )
C
C      THE METHOD EMPLOYED IS TO LOOP OVER THE DESIRED NUMBER OF
C      LEVELS ADDING AT EACH LEVEL AN X AND A Y DISPLACEMENT
C      APPROPRIATE TO THAT LEVEL AND QUADRANT.
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL*1 ISER(4096), IBLOK(64,64)
      DIMENSION I1(4),I2(4),I3(4),I4(4),I5(4),I6(4)
      DIMENSION J1(4),J2(4),J3(4),J4(4),J5(4),J6(4)
      DATA I1,I2,I3,I4,I5,I6
     1/1,-1,1,0,1,-3,1,0,1,-7,1,0,1,-15,1,0,1,-31,1,0,1,-63,1,0/
      DATA J1,J2,J3,J4,J5,J6
     1/0,1,0,0,-1,1,-1,0,-3,1,-3,0,-7,1,-7,0,-15,1,-15,0,-31,1,-31,0/
      I=1
      J=1
      N=1
C
C      LOOP OVER ALL LEVELS
C
      DO 60 L6=1,4
      DO 50 L5=1,4
      DO 40 L4=1,4
      DO 30 L3=1,4
      DO 20 L2=1,4
C
C      STORE ITEM FROM FIRST QUADRANT
C
      IBLOK(I,J)=ISER(N)
      N=N+1
C
C      STORE ITEMS FROM QUADRANTS 2 - 4
C
      DO 10 L1=1,3
      I=I+I1(L1)
      J=J+J1(L1)
      IBLOK(I,J)=ISER(N)
   10 N=N+1
      I=I+I2(L2)
   20 J=J+J2(L2)
      I=I+I3(L3)
   30 J=J+J3(L3)
      I=I+I4(L4)
   40 J=J+J4(L4)
      I=I+I5(L5)
   50 J=J+J5(L5)
      I=I+I6(L6)
   60 J=J+J6(L6)
      RETURN
      END
```

```
      SUBROUTINE GRAY3(NGRA,I128)
C
C      THIS SUBROUTINE CREATES A GRAY SCALE PLOT OF A 128 X 128
C      INPUT ARRAY ON THE PRINTER. GRAY3 OVERPRINTS UP TO TWO TIMES,
C      FOR TONES RANGING FROM WHITE (1) TO BLACK (16).
C
C      PROGRAMMER - E. MICHAEL O'NEILL
C                   COMPUTER SCIENCES CORPORATION
C                   8728 COLESVILLE RD. SILVER SPRING, MD.
C
      LOGICAL*1 I128(128,128)
      LOGICAL LS
      DIMENSION ISYM(16,3), NSYM(16), IOUT(128,3)
      EQUIVALENCE (IS,LS)
      DATA NS/16/, NL/3/, IB/1H /, IP/1H+/
      DATA ISYM/
     1 1H ,1H.,1H,,1H=,1HU,1HX,1H#,1H8,1HM,1HO,1HO,1HO,1HO,1HO,1HO,1HO,
     2 1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H=,1HI,1H+,1H*,1H*,1H*,
     3 1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H ,1H-,1HX,1HM/
      DATA NSYM /1,1,1,1,1,1,1,1,1,2,2,2,2,3,3,3/
      DO 60 J=1,128
      IPMAX=1
      DO 30 I=1,128
      LS=I128(I,J)
      IF(IS.LT.1) IS=1
      IF(IS.GT.NS) IS=NS
      DO 20 IL=1,NL
   20 IOUT(I,IL)=ISYM(IS,IL)
      IF(NSYM(IS).GT.IPMAX) IPMAX=NSYM(IS)
   30 CONTINUE
      ICC=IB
      DO 50 IL=1,IPMAX
      WRITE(NGRA,40) ICC,(IOUT(I,IL),I=1,128)
   40 FORMAT(A1,2X,128A1)
   50 ICC=IP
   60 CONTINUE
      RETURN
      END
```

REFERENCES

1. Computer Sciences Corporation, 6002-1, Organizational Structures for Constant Resolution Earth Data Bases, (prepared for the Environmental Prediction Research Facility, Monterey, California), P. R. Beaudet, F. K. Chan, and L. Goldshlak, November 1973

2. Goldstein, H., Classical Mechanics. Reading, Massachusetts: Wesley Publishing Co., June 1959